TEKNILLINEN KORKEAKOULU

Tietotekniikan osasto

Kim Hacklin

XML-pohjaisen verkkolaskun testauspalvelun suunnittelu

Diplomi-insinöörin tutkintoa varten tarkastettavaksi jätetty diplomityö, 16.1.2006

Työn valvoja: Professori Reijo Sulonen

Työn ohjaaja: Aatto J. Repo, Ph.D.

TEKNILLINEN KORKEAKOULU	DIPLOMITYÖN TIIVISTELMÄ
Tietotekniikan osasto	

Tekijä	Päiväys
Kim Hacklin	16.1.2006
	Sivumäärä
	69

Työn nimi	
XML-pohjaisen verkkolaskun testauspalvelun suunnittelu	
Professuuri	Koodi
Ohjelmistoliiketoiminta ja –tuotanto	T-76
Työn valvoja	
Professori Reijo Sulonen	
Työn ohjaaja	
Aatto J. Repo, Ph.D.	

Tutkimuksessa suunnitellaan testauspalvelu XML-pohjaiselle verkkolaskulle. Tutkimuksen tarkoitus on selvittää miten verkkolaskun testauspalvelu kannattaa toteuttaa ja miten XML-dokumenttien testaus liiketoiminnallisia sääntöjä vastaan voidaan tehdä.

Työn kirjallisuusosassa tarkastellaan sähköistä liiketoimintaa, sähköisen liiketoiminnan viitekehyksiä ja testausta sekä XML-perheen teknologioita. Lisäksi kirjallisuusosassa käydään läpi verkkolaskutusta ja vaatimusmäärittelyn sekä arkkitehtuurin suunnittelun nykytilaa.

Työn empiirisessä osassa luodaan vaatimusmäärittely, arkkitehtuurisuunnitelma sekä toiminnalliset määritykset verkkolaskun testauspalvelulle. Vaatimusmäärittely tehtiin testauspalvelun tulevien käyttäjien tarpeiden pohjalta. Arkkitehtuuri ja toiminnalliset määritykset luotiin vaatimusmäärittelyä, kirjallisuutta sekä muita vastaavia toteutuksia käyttäen. Lopuksi suoritetaan tulosten arviointi.

Tulokset osoittavat, että verkkolaskun testauspalvelu on mahdollista toteuttaa käyttäen Schematron-rakennekuvauksia ja että testauspalvelu voi ratkaista verkkolaskutuksen teknisiä ongelmia. Verkkolaskutuksen suurimpana haasteena ovat kuitenkin liiketoiminnalliset ongelmat. Nämä haasteet ja sähköisen liiketoiminnan testaaminen yleisesti nostivat esille kysymyksiä jatkoa varten.

Avainsanat

Verkkolasku, sähköisen liiketoiminnan testaus, yhteentoimivuus, Schematron

HELSINKI UNIVERSITY OF TECHNOLOGY ABSTRACT OF MASTER'S THESIS Department of Computer Science and Engineering	
Author Kim Hacklin	Date 16.1.2006 Pages 69
Title of thesis Designing a validation service for XML-based electronic invoices	
Professorship Software Business and Engineering	Professorship Code T-76
Supervisor Professor Reijo Sulonen	
Instructor Aatto J. Repo, Ph.D.	

This thesis is used to investigate how to design a validation service for XML-based electronic invoices. The purpose of the thesis is to find out the optimal design for such a service and to look into the validation of XML documents based on business rules.

The literature study consists of an analysis of ebusiness in general, ebusiness frameworks, testing ebusiness solutions and the most important XML technologies. Also the current state of electronic invoicing, requirements engineering and architectural design are investigated.

In the empirical part of this thesis, the requirements and architecture for the validation service were created. The requirements were based on the needs of the future users of the validation service The architecture and functional requirements were built using the requirements, the literature study and existing implementations of a validation service. Finally, the results of the empirical part are evaluated.

The results show that a validation service for electronic invoices can be built based on Schematron schemas and that such a service can be used to solve technical problems in electronic invoicing. However, the biggest challenges come from the business side. These challenges together with open issues in ebusiness testing gave rise to several questions to be solved later.

Keywords

Electronic invoice, ebusiness testing, interoperability, Schematron

Alkusanat

Tämä diplomityö toteutettiin Tietoyhteiskunnan kehittämiskeskus ry:lle (TIEKE). Tavoitteena oli suunnitella verkkolaskun testauspalvelu verkkolaskutuksen teknisten ongelmien ratkaisemiseksi. Haluan kiittää TIEKEn toiminnanjohtajaa, ohjaajaani Aatto J. Repoa mahdollisuudesta olla mukana verkkolaskutuksen kehityksen kärjessä ja käsitellä haastavaa ja mielenkiintoista aihetta. Kiitän myös valvojaani Reijo "Shosta" Sulosta työni ohjauksesta ja kommentoimisesta.

Suuri kiitos kuuluu myös Timo Simellille työni tarkastamisesta ja arvokkaista vihjeistä kirjoittamisen aikana. Ystävälleni Juha Ikävalkolle tahdon yksinkertaisesti sanoa kiitos, sillä ilman sinua tämän työn tekeminen ei olisi päässyt alkuun, saati ikinä valmistunut.

Eniten haluan kiittää perhettäni, jota ilman en olisi opinnoistani selvinnyt. Kiitos vanhemmilleni kaikesta tuesta näiden lukuisten vuosien varrella. Ja ennen kaikkea kiitos sisaruksilleni Kiralle ja Kristianille siitä, että olette olemassa.

Helsingissä, 16.1.2006,

Kim Hacklin

1 Johdanto

1.1 Taustaa

Sähköinen liiketoiminta on yhä useammalle yritykselle keskeinen osa liiketoimintaa. Sähköisellä liiketoiminnalla tarkoitetaan Internetin käyttöä yrityksen asiakkaiden, toimittajien sekä sisäisten ja ulkoisten järjestelmien yhdistämiseksi (Rodgers, Yen & Chou 2002). Sähköisen liiketoiminnan avulla yritykset voivat pienentää kulujaan, vähentää virhetilanteiden määrää ja nopeuttaa tiedonkulkua. Internetin nousu maailmanlaajuiseksi tietoverkoksi ja XML-metakieli ovat mahdollistaneet monipuolisten sähköisen liiketoiminnan viitekehysten synnyn.

Sähköisen liiketoiminnan viitekehykset sisältävät määrityksiä sanomista, protokollista ja prosesseista. Viitekehysten tehokas käyttö yrityksissä edellyttää, että yrityksen tietojärjestelmät toteuttavat kaikkia viitekehysten osia määritysten mukaisesti. Määritykset ovat monimutkaisia, mutta toisaalta ne eivät ole aina riittävän yksiselitteisiä. Näin ollen kaksi eri tietojärjestelmää todennäköisesti toteuttaa määrityksiä hieman eri tavalla ja tästä syntyy yhteentoimivuusongelmia. Ongelmien ratkaisemiseksi järjestelmien yhteentoimivuutta täytyy testata, ja viitekehyksille on olemassa erilaisia testausalustoja ja työkaluja yhteentoimivuuden testaamiseksi. Yhteentoimivuuden testaaminen on välttämätön, mutta ei riittävä edellytys sähköisen liiketoiminnan viitekehysten onnistuneelle toiminnalle.

Laskutus on yksi liiketoiminnan ja samalla sähköisen liiketoiminnan keskeisimmistä prosesseista. Sähköisen laskutuksen sanomaa kutsutaan sähköiseksi laskuksi tai verkkolaskuksi. Verkkolasku on lasku, joka lähetetään ja vastaanotetaan koneellisesti ja joka voidaan helposti tulostaa ruudulle katseltavaan muotoon (TIEKE 2005). Verkkolaskun käytön esteitä ovat tällä hetkellä mm. erilaiset yhteensopivuusongelmat, verkkolaskuformaattien sovellusohjeiden tulkinnanvaraisuus, yleinen toimintavarmuuden puute käyttäjien näkökulmasta sekä alan toimijoiden liiketoimintamallien erilaisuudet. Tekniset ongelmat pystytään poistamaan testaamalla verkkolaskuohjelmistoja ja verkkolaskun kulkua, mutta verkkolaskuformaatteja ja – toimijoita on niin paljon, että kaikkien toimijoiden välinen testaus ei ole taloudellisesti järkevää. Tehokkaampi tapa on käyttää keskitettyä testauspalvelua.

Tämän tutkimuksen tavoite on keskitetyn verkkolaskun testauspalvelun suunnittelu tekemällä palvelun vaatimusmäärittely ja arkkitehtuurisuunnitelma. Tutkimuksessa keskitytään sähköisen liiketoiminnan yhteentoimivuusongelmiin ja näiden ongelmien ratkaisemiseen

testaamalla. Testaamisen osalta käsitellään erityisesti XML-muotoisten sanomien validointia käyttäen rakennekuvauskieliä jotka ovat ilmaisuvoimaisempia kuin W3C:n XML Schema. Tutkimuksessa käsitellään myös vaatimusmäärittelyn ja arkkitehtuurisuunnittelun nykytilaa, yleisesti suositeltavia toimintatapoja ja käsiteltävään ongelmaan soveltuvia menetelmiä. Vaatimusmäärittely tehdään sidosryhmien tarpeiden perusteella ja arkkitehtuuri suunnitellaan vaatimusmäärittelyn tuloksien pohjalta.

Testauspalvelulla testataan verkkolaskujen tietosisällön oikeellisuutta. Testaussääntöjen määrittelyssä käytetään verkkolaskuformaattien soveltamisohjeita sekä kokemuksia verkkolaskujen käytännön ongelmista.

Tutkimuksen käytännön osuus tehtiin maaliskuusta kesäkuuhun vuonna 2005. Ongelman määrittely ja tutkimuksen tavoitteet tulivat toimeksiantajalta TIEKEltä (Tietoyhteiskunnan kehittämiskeskus ry). Verkkolaskutuksen keskeiset toimijat Suomessa eli verkkolaskuoperaattorit ja pankit perustivat yhdessä TIEKEn kanssa verkkolaskufoorumin syksyllä 2001. Verkkolaskufoorumin tavoitteena on edistää verkkolaskun käyttöä. Foorumi tunnisti ongelmat verkkolaskun käytössä ja päätti, että keskitetty testauspalvelu mahdollistaisi kattavan testauksen kustannustehokkaasti. Testauksessa päätettiin keskittyä verkkolaskun lähetykseen ja vastaanottoon eli verkkolaskuohjelmistoihin. Verkkolaskun testauspalvelun ensisijaisia käyttäjiä olisivat siten verkkolaskufoorumin ohjelmistotalot. Nämä ohjelmistotalot valmistavat sähköisen taloushallinnon järjestelmiä jotka lähettävät ja/tai vastaanottavat verkkolaskuja.

1.2 Tutkimusongelma ja – tavoitteet

Tämän tutkimuksen tutkimusongelma on seuraava: Miten XML – muotoisen verkkolaskun testauspalvelu tulisi suunnitella? Suunnittelun reunaehdot liittyvät lähtötilanteeseen eli verkkolaskutuksen tilaan Suomessa vuoden 2005 keväällä. Tutkimuksessa haetaan lisäksi vastauksia seuraaviin kysymyksiin:

- Miten XML - dokumentin sisältöä voidaan testata liiketoiminnallisia sääntöjä vastaan?
- Miten sähköisen liiketoiminnan asiakirjoja ja prosesseja yleisesti ottaen voi testata?
- Minkälaisia vastaavia toteutuksia sähköisen liiketoiminnan ja verkkolaskun testauspalveluista on olemassa?
- Mitkä ovat teknisen testaustyökalun mahdollisuudet ratkaista liiketoiminnallisia ongelmia verkkolaskutuksen nykytilassa?

Verkkolaskun testauspalvelun suunnittelu kattaa tässä tutkimuksessa vaatimusmäärittelyn, arkkitehtuurin suunnittelun ja toiminnalliset määritykset.

1.3 Tutkimusmenetelmät

Tutkimuksessa tarkastellaan ensin sähköisen liiketoiminnan viitekehysten ja testauksen kirjallisuutta. Sähköisestä liiketoiminnasta löytyy lukuisia hyviä teoksia, mutta sähköisen liiketoiminnan testaus on uusi asia joka on vasta nyt saamassa enemmän huomiota. Tämän takia sähköisen liiketoiminnan testauksesta löytyy varsin niukasti kirjallisuusviitteitä. Tutkimuksessa käydään läpi esimerkkitoteutuksia laajemman ymmärryksen saavuttamiseksi.

Nykytilan kartoituksen jälkeen luodaan vaatimusmäärittely verkkolaskun testauspalvelusta. Vaatimusmäärittelyn tiedonkeruu tapahtuu pääasiassa haastatteluiden ja tapaamisten kautta. Vaatimusmäärittelyn pohjalta suunnitellaan testauspalvelun arkkitehtuuri sekä toiminnalliset määritykset. Arkkitehtuurin suunnittelussa käytetään tutkimuksen ensimmäisen osan esimerkkitoteutuksia hyväksi.

Konstruktiivisen osuuden tuloksia arvioidaan läpikäynneissä sidosryhmien kanssa sekä itsenäisesti suunnitteluprosessien onnistumisen osalta.

1.4 Laajuus ja rajaukset

Tutkimuksessa käsitellään verkkolaskun testauspalvelun vaatimusmäärittelyä, arkkitehtuurin suunnittelua sekä toiminnallista määrittelyä. Ohjelmistotuotannon muut vaiheet kuten yksityiskohtainen suunnittelu, toteutus ja projektinhallinta on rajattu tutkimuksesta kokonaan pois. Vaatimusmäärittelyn osalta tutkimuksesta on jätetty pois vaatimustenhallinta eli vaatimusten muutosten käsittely varsinaisen vaatimustenmäärittelyn jälkeen.

Tutkimuksessa keskitytään pääasiassa teknisten ongelmien ratkaisemiseen. Tutkimuksen aikana havaittiin, että teknisten ongelmien taustalla on erilaisia liiketoiminnallisia ongelmia kuten alan toimijoiden liiketoimintamallien erot ja erilaiset lähestymistavat verkkolaskutukseen. Tutkimuksen pääpaino on kuitenkin teknisten ongelmien ratkaisussa.

Tutkimuksessa etsitään vastaavia toteutuksia ja ratkaisuja ja mahdollisuuksien mukaan käytetään näitä tutkimuksen pohjana.

Tutkimuksessa on tehty vaatimusmäärittely, arkkitehtuurisuunnitelma ja toiminnalliset määritykset. Tutkimus on kirjoitettu sovelluskehittäjille jotka ovat tekemisissä XML-sanomien (erityisesti verkkolaskujen) testaamisen kanssa.

1.5 Rakenne

Tämä tutkimus koostuu seuraavista osista:

Johdanto. Johdannossa esitellään tutkimuksen tavoite, tutkimusongelma, tutkimusmetodologia, tutkimuksen laajuus ja rajaukset sekä rakenne.

Nykytilan kartoitus. Osiossa käydään läpi sähköistä liiketoiminta yleisesti verkkolaskun osalta ja esitellään keskeisiä teknologioita (XML, rakennekuvaukset, testaus, sähköinen liiketoiminta). Lisäksi keskitytään erityisesti sähköisen liiketoiminnan sanomien testaukseen. Kartoituksessa käydään myös läpi vaatimusmäärittelyn ja arkkitehtuurisuunnittelun nykytilaa.

Tulokset. Tuloksissa esitellään testauspalvelun vaatimusmäärittely ja arkkitehtuuri sekä testauspalvelun toimintaperiaate toiminnallisten määritysten kautta.

Tulosten tarkastelu. Tarkastellaan tutkimuksen tuloksia ja käydään keskustelua tutkimuksen aikana esille tulleista avoimista kysymyksistä sekä mahdollisista jatkotutkimuskohteista.

Johtopäätökset ja yhteenveto. Esitellään tutkimuksen johtopäätökset sekä yhteenveto.

2 Sähköinen liiketoiminta

Luvussa esitellään sähköinen liiketoiminta ja sähköisen liiketoiminnan viitekehykset ebXML, RosettaNet ja EDI.

2.1 Yleistä

Sähköisellä liiketoiminnalla tarkoitetaan yleisesti Internetin avulla tehtävää liiketoimintaa. Tietotekniikan käyttö liiketoiminnassa on kuitenkin niin laajalle levinnyttä, että ei ole aina edes mielekästä puhua erikseen sähköisestä liiketoiminnasta. Tämä näkyy termin määrittämisen vaikeudessa. Sähköisellä liiketoiminnalla tarkoitetaan yrityksen sisäisten ja ulkoisten toimintojen suunnittelua ja toteuttamista Internetin avulla (Lee & Whang 2001) tai laajempaa kokonaisuutta johon kuuluu sekä kaupallisia että ei-kaupallisia toimintoja (Li 2003). Sähköisen liiketoiminnan (e-business) erottaminen sähköisestä kaupankäynnistä (e-commerce) on myös hankalaa. Useimmiten sähköinen liiketoiminta nähdään kattavampana käsitteenä kuin sähköinen kaupankäynti (Li 2003) (Rodgers, Yen & Chou 2002). Lisäksi on olemassa käsite *yritystenvälinen sähköinen kaupankäynti* (B2B e-commerce) joka tarkoittaa yritystenvälistä tai – sisäistä rakenteellisen tiedon vaihtoa sähköisessä kaupankäynnissä (Lim & Wen 2002).

Oleellista sähköisessä liiketoiminnassa on mahdollisuus lähettää, vastaanottaa ja käsitellä tietoa koneellisesti. Näin voidaan nopeuttaa tiedonkulkua, vähentää virheiden määrää ja leikata kustannuksia (Lim & Wen 2002). Tämä yhdessä sen kanssa, että sähköinen liiketoiminta on muuttumassa kilpailuedusta välttämättömyydeksi, selittää yritysten kiinnostuksen aiheeseen. Odotukset kasaantuvat erityisesti pienten ja keskisuurten yritysten kohdalle, sillä näiden keskuudessa sähköisen liiketoiminnan hyödyntäminen on vielä alhaisempaa kuin suurten yritysten kohdalla (Statistics Denmark et al.).

2.2 Sähköisen liiketoiminnan viitekehykset

Sähköisen liiketoiminnan viitekehykset ovat yleisiä toimintamalleja joiden avulla yritykset voivat varmistaa yhteentoimivuuden sähköisessä liiketoiminnassa (Shim et al. 2000). Viitekehysten avulla pyritään ratkaisemaan ongelmia, jotka syntyvät yritysten erilaisten järjestelmien ja liiketoimintaprosessien yhteensovittamisesta. Sähköisen liiketoiminnan viitekehyksissä määritellään yritysten välillä vaihdettavien asiakirjojen rakenne, asiakirjojen välitystapa sekä ulkoiset liiketoimintaprosessit, joiden mukaan asiakirjoja vaihdetaan.

Seuraavaksi tehdään lyhyt katsaus kolmeen yleisimpään sähköisen liiketoiminnan viitekehykseen, Electronic Data Interchange (EDI), RosettaNet ja ebXML. RosettaNet ja ebXML ovat näistä viitekehyksistä uudempia ja täysin XML-pohjaisia. EDI on ollut käytössä pidempään ja poikkeaa teknisesti kahdesta muusta viitekehyksestä. EDI on myös nykyisin eniten käytössä oleva viitekehys sen pitkästä taustasta johtuen. Verkkolasku liittyy kiinteästi sähköisen liiketoiminnan viitekehyksiin, sillä verkkolaskussa on käytetty viitekehysten ominaisuuksia ja elementtejä. Lisäksi näiden viitekehysten testausmenetelmiä voidaan soveltaa sellaisenaan myös verkkolaskun testaukseen.

2.2.1 RosettaNet

RosettaNet on sähköisen liiketoiminnan viitekehys, joka on luotu pääosin yritysten muodostaman kansainvälisen yhteenliittymän toimesta (RosettaNet 2005). RosettaNet:n kehitys alkoi vuonna 1998 elektroniikkateollisuuden tarpeista, ja vaikka se on viime vuosina laajentunut muillekin teollisuudenaloille, käytetään RosettaNet:iä eniten juuri elektroniikka- ja tietotekniikkayrityksissä.

RosettaNet koostuu kolmesta eri osasta: liiketoimintasanastosta, liiketoimintaprosesseista ja sanomien välityksestä (RosettaNet 2005). *Liiketoimintasanastossa* määritellään yhteinen termistö sanomien muodostamista varten. Termistön avulla kuvataan liiketoimintaprosessit sekä tuotetietoja. *Liiketoimintaprosessit* (Partner Interface Process, PIP) määrittävät välitettävät sähköiset asiakirjat sekä niiden järjestyksen kauppakumppanien välillä.

RosettaNet on tässä tutkimuksessa mielenkiintoinen, koska sille on olemassa testausalustoja ja -menetelmiä, joita voidaan hyödyntää tässä suunniteltavassa testauspalvelussa.

2.2.2 ebXML

ebXML on YK:n (UN/CEFACT) ja OASIS:n (Organization for the Advancement of Structured Information Standards) aloitteesta lähtenyt standardisointityö johon on liittynyt mukaan monia yrityksiä ja yhteisöjä. EbXML:n tavoitteena on kehittää avoin ja globaali sähköisen liiketoiminnan viitekehys.

EbXML on varsinaisesti ryhmä standardeja johon kuuluu mm. viestintäkerros ebMS (ebXML Messaging Services), prosessimäärittelyt (ebXML Business Process Specification Schema) sekä tietorekisteri (ebXML Registry).

EbXML:ää käytetään Finvoice-verkkolaskuformaatissa. Lisäksi ebXML:lle löytyy perusteellisesti määritelty testausalusta joista on myös useita toteutuksia. Suurin osa

sähköisen liiketoiminnan testauksen kirjallisuudesta käsittelee nimenomaan ebXML:n testausta.

2.2.3 EDI

EDI (Electronic Data Interchange) tarkoittaa yleisesti tietokoneiden välistä sähköistä tiedonsiirtoa. Tässä EDI:llä tarkoitetaan UN/EDIFACT:a joka on yksi laajimmin käytetyistä EDI-standardeista. EDI on nykyisten XML-pohjaisten viitekehysten edeltäjä joka alunperin käytti varta vasten rakennettuja yhteyksiä siirtämään liiketoiminnallista tietoa yritysten välillä.

Suomessa on käytetty EDI:ä 1980-luvun alkupuolesta lähtien ja se on saavuttanut vahvan aseman nimenomaan suurten yritysten välisessä tiedonsiirrossa. EDI on käyttökelpoinen vielä pitkään tulevaisuudessa, mutta sen heikkoudet estävät sen leviämisen laajempaan käyttöön. EDIn implementointi on kallista ja se vaatii eritysasiantuntijoita jolloin useimmilla pk-yrityksillä ei ole mahdollisuutta ottaa EDI:ä käyttöön.

Nykyinen verkkolasku pohjautuu voimakkaasti EDIin ja verkkolaskutuksen toimijoilla on usein pitkä tausta EDI-liikenteen käsittelyssä. Lisäksi verkkolaskutuksen liiketoimintamallit tulevat pitkälti EDI-maailmasta.

3 Sähköinen laskutus ja verkkolasku

Luvussa esitellään sähköinen laskutus ja verkkolasku. Verkkolaskusta esitellään nykytila, käytännöt Suomessa, erilaiset verkkolaskuformaatit sekä verkkolaskun välitysketju.

3.1 Sähköinen laskutus

Yrityksen laskutusprosessit voidaan jakaa osto- ja myyntilaskutukseen sekä yrityksen sisäisiin ja ulkoisiin prosesseihin. Laskutusprosessien ymmärtäminen on välttämätöntä verkkolaskujen käytön ja ongelmien ymmärtämiseksi. Tässä tutkimuksessa keskitytään ostolaskutukseen ja yrityksen ulkoisiin prosesseihin. Tutkimuksen kannalta tärkeitä osia laskutusprosessista ovat laskun muodostaminen, lähetys, vastaanotto ja sähköinen käsittely. Tutkimuksessa ei oteta kantaa esim. laskun maksamiseen tai kirjanpidon ja laskutuksen kytkemiseen toisiinsa. Tutkimuksessa keskitytään myös enemmän yksittäisiin laskuihin kuin koko laskutusprosessiin.

Sähköinen laskutus liittyy usein kiinteästi yrityksen taloushallinnon ja muiden liiketoiminnan osa-alueiden sähköistämiseen. Monien näiden muutosten taustalla on Suomessa tehty kirjanpitolain muutos, joka mahdollisti kirjanpitomateriaalin sähköisen arkistoinnin riippumatta siitä onko alkuperäinen materiaali sähköistä vai paperilla. Tämä muutoksen johdosta yritysten kannatti ottaa käyttöön sähköisiä laskujen kierrätysjärjestelmiä, sähköinen arkistointi sekä sähköinen laskutus.

Yritykset siirtyvät sähköiseen laskutukseen usein vaiheittain. Yrityksissä on usein sähköistetty sisäisiä laskutusprosesseja, mutta kahden yrityksen, ostajan ja myyjän, välillä lasku kulkee yhä useimmiten paperilla. Tämän tilanteen ratkaisemiseksi on syntynyt laskujen tulostus- ja skannauspalveluita. Näiden avulla yritykset voivat pitää sisäiset laskutusprosessinsa sähköisinä ja samalla lähettää ja/tai vastaanottaa paperisia laskuja sellaisilta kumppaneilta, jotka eivät kykene käsittelemään sähköisiä laskuja.

3.2 Verkkolasku

3.2.1 Yleistä

Verkkolasku on "sähköinen lasku, jonka tiedot ovat automaattisesti käsiteltävissä ja josta voidaan tuottaa tietokoneen näytölle paperilaskua muistuttava näkymä" (TIEKE 2005a). Verkkolaskun erottaa muista sähköisesti välitettävistä laskuista kaksi asiaa: automaattinen käsittely ja näkymän muodostaminen laskusta näytölle. Sähköpostitse välitettävä PDF-

muotoinen lasku on sähköinen lasku, mutta ei verkkolasku, koska sitä ei voida automaattisesti käsitellä. EDI-laskusta taas ei voi muodostaa helposti näkymää tietokoneen näytölle, joten se ei myöskään ole verkkolasku (Sisäasiainministeriö 2003). Tässä tutkimuksessa käsitellään suomalaista XML-pohjaista verkkolaskua. Näin ollen tutkimuksen ulkopuolelle jäävät muut kuin XML-pohjaiset verkkolaskut sekä muut sähköiset laskut, jotka eivät ole verkkolaskuja. Verkkolaskun määritelmä ei ole kuitenkaan täysin vakiintunut ja sen erottaminen muista sähköisistä laskuista on osittain keinotekoista. Verkkolasku-termin voidaan katsoa olevan enemmänkin markkinointitermi uusille XML-pohjaisille laskuformaateille.

Kansainvälisesti sähköinen lasku (electronic invoice, einvoice) tarkoittaa useimmiten EDI-laskua, mutta verkkolasku sellaisena kuin se Suomessa käsitetään, on myös tulossa laajempaan käyttöön. Yritysten lisäksi myös eri maiden julkishallinnot ovat ryhtyneet edistämään verkkolaskun käyttöä ja pohjoismaat ovat olleet tässä edelläkävijöitä. Tanskassa julkinen sektori siirtyi käyttämään laskutuksessa yksinomaan verkkolaskuja 1.2.2005 alkaen (OES 2005). Suomessa sisäasiainministeriö julkaisi jo vuonna 2003 suosituksen verkkolaskujen käytöstä julkishallinnossa (Sisäasiainministeriö 2003). Suosituksessa rohkaistaan julkishallintoa ottamaan verkkolaskut käyttöön.

Verkkolaskun edut ovat samat kuin sähköisessä liiketoiminnassa yleensä eli sillä voidaan nopeuttaa tiedonkulkua ja vähentää virheiden määrää. Suurimpana houkuttimena yrityksille verkkolaskun käytössä on kuitenkin suora rahallinen säästö, koska verkkolaskun käsittelyssä on vähemmän manuaalisia vaiheita kuin paperilaskun käsittelyssä. Verkkolaskun käsittely sujuu nopeammin ja halvemmalla kuin paperilaskun. Alla on esitetty paperilaskun ja verkkolaskun käsittelyyn vaadittava aika ja kustannukset (Vahtera & Salmi 1997). Esimerkki on tehty suurten yritysten näkökulmasta ja ajalliset säästöt ovat pienempiä pk-yrityksille:

Käsittelyvaihe	Paperilasku, aika (min)	Verkkolasku, aika (min)
Postin avaaminen	1	
Lyödään päivämääräleima laskulle	1	
Otetaan kopio originaalista	1	
Kopio mappiin aakkosjärjestykseen	1	
Tarkastus ja tiliöinti (laskulle)	2	
Syöttö ostoreskontraan	2	
Asiatarkastus	1	1
Hyväksyminen	2	1
Laskun tiliöinti tietojärjestelmään	1,5	
Hyväksyminen maksuun	0,5	
Laskun arkistointi (numerojärjestys)	1	

In-house postitus (9 kopiota laskusta)	10	
Virheiden käsittely (10% laskuista)	2	1
Yhteensä	**26**	**3**
Työtunnin hinta = 34 EUR		
Työn kustannus / lasku	14,6 EUR	1,7 EUR
Säästö prosentteina		**88,5 %**

Taulukko 1: Paperilaskun ja verkkolaskun käsittelyn vaiheet ja kustannukset (Vahtera & Salmi 1997)

Valtiokonttori on tehnyt arvion verkkolaskutuksen säästöistä, jonka mukaan verkkolasku on noin 30 euroa halvempi kuin paperilasku (Valtiokonttori 2005). Valtiokonttori käsittelee 4 miljoonaa ostolaskua vuodessa joten verkkolaskun käyttö voi mahdollistaa 120 miljoonan euron säästöt veronmaksajille.

Verkkolaskujen käytön määrä on hankala arvioida, mutta Suomi on joka tapauksessa verkkolaskun osalta kansainvälisen kehityksen kärjessä. Suomessa lähetetään vuosittain 150-250 miljoonaa laskua yritysten välillä (IDC 2003) (Posti 2003) (Market-Visio 2005). Eri tutkimusten mukaan verkkolaskujen määrä kaikista yritystenvälisistä laskuista oli 3 – 4 % vuonna 2003 (IDC 2003) (Posti 2003) ja vuonna 2005 9% (Market-Visio 2005). Verkkolaskujen määrän kasvu arvioidaan erittäin voimakkaaksi ja vuonna 2007 verkkolaskuja voisi Suomessa olla jo yli 30% kaikista yritystenvälisistä laskuista (IDC 2003).

3.2.2 Verkkolaskujen välittäminen

Suomessa verkkolaskuja välittävät operaattorit (joista käytetään myös termiä verkkolaskuoperaattori) ja pankit. Operaattoreista ja pankeista käytetään tässä tutkimuksessa yhteisnimitystä *välittäjät*. Välittäjien toimintatavoissa on eroja jotka johtuvat erilaisista liiketoimintamalleista, kilpailutilanteesta, erilaisista lähtökohdista sekä verkkolaskutuksen nopeasta kasvusta ja muutoksesta.

Pankkien näkökulmasta verkkolasku on sähköisen maksamisen ja sähköisen pankkitoiminnan luonnollinen jatke. Pankeilla ei ole perinteisesti ollut roolia laskujen välittäjänä vaan pankit ovat hoitaneet laskun maksamiseen liittyvät tehtävät. Suomessa suurimmalla osalla yrityksistä on käytössä pankkiyhteysohjelma jonka avulla yritykset voivat automatisoida laskujen maksamista. Pankkiyhteysohjelma kerää yrityksen maksutapahtumat ja lähettää ne pankkiin maksettavaksi koneellisessa muodossa. Suomessa yritysten maksuliikenne on pitkään ollut lähes täysin sähköistä ja verkkopankin käyttö on erittäin yleistä.

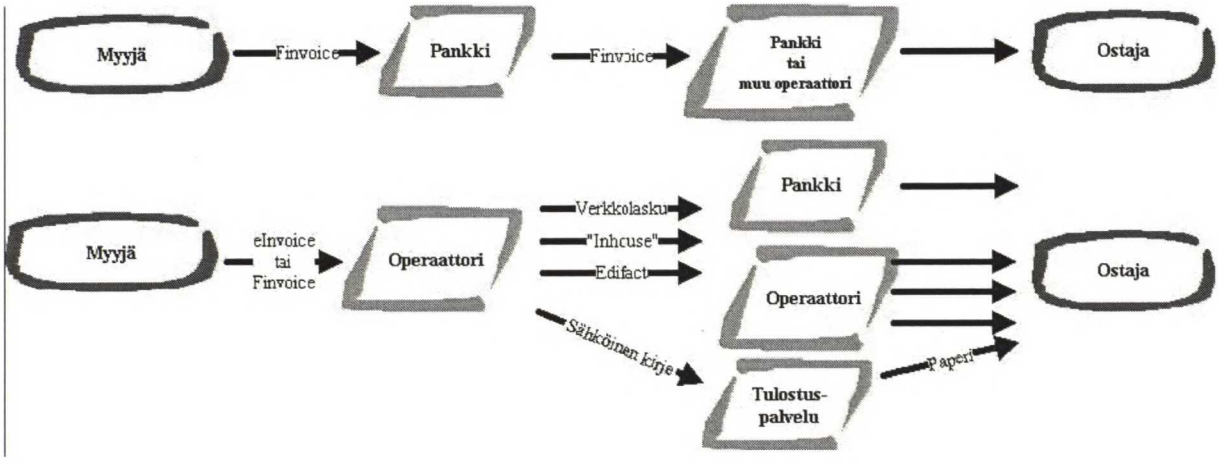
Operaattorien näkökulmasta taas verkkolasku liittyy yritysten liiketoimintaprosessien sähköistämiseen osana sähköistä taloushallintoa. Verkkolasku on tällöin yksi sähköinen

sanoma muiden (tarjous, tilaus, tilausvahvistus, toimitusluettelo) joukossa. Verkkolaskun välittäminen voidaan rinnastaa paperisenlaskun välittämiseen postitse.

Pankit keskittyvät vain verkkolaskun välitykseen. Operaattorit tarjoavat verkkolaskun välityksen lisäksi muita palveluita kuten laskujen tulostusta ja skannausta sekä muunnoksia verkkolaskuformaatista toiseen. Keskeisimmät Suomessa käytössä olevat verkkolaskuformaatit ovat Finvoice, eInvoice ja TEAPPSXML. Lisäksi monilla, varsinkin suuremmilla yrityksillä, on käytössä sisäisiä "inhouse" - laskuformaatteja.

Verkkolaskun välittäminen tarkoittaa laskun välittämistä myyjältä ostajalle. Välityksessä on kaksi vaihtoehtoa (kuva 1):

1. Jos myyjä lähettää verkkolaskun pankille, lähettää pankki laskun eteenpäin toiselle pankille tai operaattorille, joka taas lähettää laskun edelleen ostajalle. Lasku kulkee myyjältä pankille ja pankilta edelleen toiselle pankille tai operaattorille Finvoice-verkkolaskuformaatin mukaisessa muodossa. Mikäli toinen välittäjä on operaattori, voi se tarjota mahdollisuuden muuntaa lasku Finvoice -formaatista johonkin toiseen formaattiin.
2. Jos myyjä lähettää verkkolaskun operaattorille, lähettää operaattori laskun eteenpäin pankille tai toiselle operaattorille, joka välittää laskun edelleen ostajalle. Lisäksi operaattori voi lähettää laskun tulostuspalveluun, josta se voidaan lähettää eteenpäin paperisena laskuna ostajalle. Lasku kulkee myyjältä operaattorille Finvoice, eInvoice tai muussa muodossa, josta operaattori tarpeen mukaan muuntaa laskun toiseen formaattiin.



Kuva 1: Verkkolaskun lähettäminen ja vastaanotto (Sisäasiainministeriö 2003)

Kun myyjä haluaa lähettää verkkolaskun ostajalle, täytyy kummallakin taholla olla verkkolaskusopimus joko pankin tai operaattorin kanssa. Pankeilla ja operaattoreilla on omat verkostonsa, joissa ne ovat tehneet lisäksi sisäisiä sopimuksia verkkolaskujen välityksestä (kuva operaattori- ja pankkiverkostoista). Suomessa onkin kaksi erilaista tapaa verkkolaskujen välitykseen: pankkien Finvoice-välityspalvelu pankkiverkon kautta sekä operaattorien yhdysliikennesopimuksiin perustuva välittäminen.

Verkkolaskut välitetään yleensä eräajoina eli laskuja lähetetään ja vastaanotetaan esimerkiksi kerran päivässä. Tämä käytäntö on perua sekä EDI:n käytöstä että pankkiyhteysohjelmista jotka toimivat samalla eräajoperiaatteella. Verkkolaskujen välitys tapahtuu useimmiten käyttäen FTP-protokollaa .

Jokaisella myyjällä ja ostajalla, joka haluaa lähettää ja/tai vastaanottaa verkkolaskuja, on oltava *verkkolaskuosoite*. Verkkolaskuosoitteen voidaan katsoa olevan normaalin laskutusosoitteen sähköinen vastine. Lisäksi välittäjillä on omat verkkolaskuosoitteensa. Yrityksellä voi olla yksi tai useampi verkkolaskuosoite. Verkkolaskuosoitteita on kolmea eri tyyppiä:

OVT-tunnus eli organisaatioiden välisen tiedonsiirron osapuolitunnus on määritelty SFS 5748 – standardissa. OVT-tunnus on muotoa 0037YYYYYYYYNNNNN, missä 0037 on Suomen verohallinnon koodi, YYYYYYYY on yrityksen Y-tunnus ja NNNNN on yhteisön osan tunnus jota käytetään erottamaan toisistaan saman yrityksen eri yksiköt (Elma 2005a).

Esimerkiksi Ruukki Group Oyj:n OVT-tunnus on 003706181818 (TIEKE 2005c).

Verkkolaskutilitunniste on suomalaisten operaattorien käyttämän verkkolaskuosoite. Tunniste on muotoa BPPTFINNNNNNNN, missä PP on operaattorin tunnus, T on tilin tyyppi ja NNNNNNNN on varsinainen osoite (eInvoice Consortium 2005).

Esimerkiksi Ruukki Group Oyj:n verkkolaskutilitunniste on BELRFI00003555 (TIEKE 2005c).

IBAN on pankkien käyttämä kansainvälinen tilinumero. Se on muotoa FITTNNNNNNNNNNNN, missä TT on tarkiste ja NNNNNNNNNNNNN on yrityksen suomalainen tilinumero (Elma 2005a).

Esimerkiksi Oulun energian IBAN on FI7115853000020003 (TIEKE 2005c).

Verkkolaskuosoitetta tarvitaan laskun *reitittämiseen* myyjältä ostajalle välittäjien kautta. Verkkolaskuformaattien ja välittäjien reitityskäytännöt eroavat toisistaan siten, että

verkkolaskuja voidaan välittää suoraan vastaanottajan osoitteen perusteella tai välittäjän osoitteen perusteella.

3.2.3 Verkkolaskuformaatit

Verkkolaskuformaatti määrittelee verkkolaskun rakenteen ja käyttötavan. Verkkolaskuformaattiin sisältyy usein rakennekuvaus ja soveltamisohje, jotka määrittelevät verkkolaskuformaatin mukaisen laskun.

Suomessa on käytössä useita erilaisia verkkolaskuformaatteja. Jotkut formaatit ovat syntyneet eri toimijoiden yhteenliittymän kautta, mutta myöhemmin yhteenliittymä on purettu tai formaatin kehittämisvastuu on siirtynyt vain yhdelle toimijalle. Muuttuneet tarpeet ja kehittynyt teknologia ovat synnyttäneet erilaisia formaatteja ja myös muokanneet olemassa olevia formaatteja. Rinnakkaisten standardien yhteiselo on mahdollista, mutta usein joku standardeista saa hallitsevan aseman ja voi päätyä ainoaksi käytössä olevaksi standardiksi. Yritys tai taho, joka omistaa hallitsevan standardin on edullisessa asemassa kilpailijoihin nähden. Tästä johtuen yritykset pyrkivät luonnollisesti edistämään omaa verkkolaskuformaattiaan.

Verkkolaskuformaatilla on siis useimmiten haltija, joka koordinoi formaatin kehitystä. Alla on listattu suurimmat Suomessa käytössä olevat verkkolaskuformaatit ja niiden haltijat:

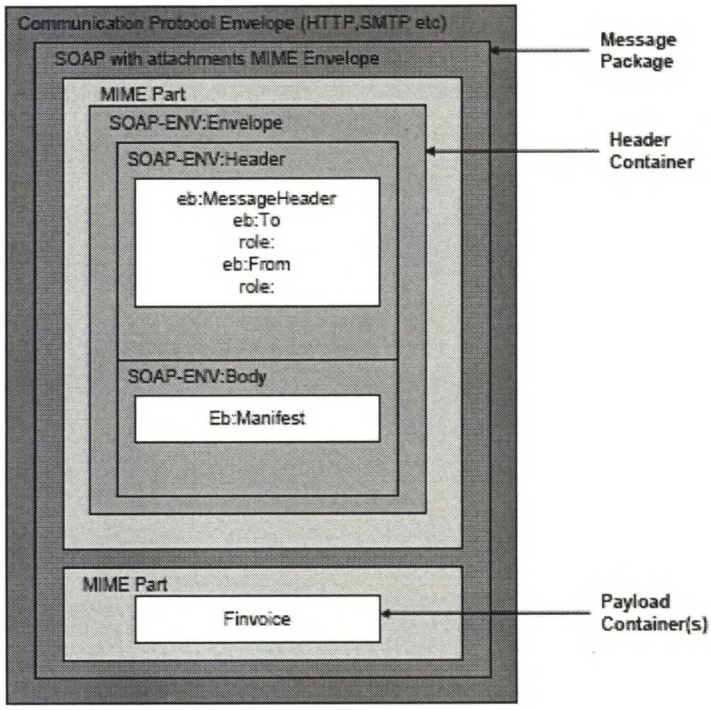
- Finvoice (Suomen Pankkiyhdistys)
- eInvoice (Pohjoismainen verkkolaskukonsortio)
- TEAPPSXML (TietoEnator)

Verkkolaskun rakenne on yleisellä tasolla sama formaatista riippumatta. Kaikissa verkkolaskuissa on pohjatiedot joissa kerrotaan mm. laskun lähettäjän ja vastaanottajan tiedot, laskun päivämäärä sekä muuta yleistä tietoa. Tämän jälkeen laskussa on laskurivejä, joilla kerrotaan laskutettavan tuotteen tai palvelun tiedot. Lisäksi laskulla voi olla ympäröivä kehysrakenne laskun reitittämistä varten.

Verkkolaskussa pitää olla vähintään ne tiedot, jotka ALV-laki edellyttää paperilaskuiltakin. Pienimmässä verkkolaskussa voi olla 24 tietoelementtiä (TIEKE 2005d), mutta käytännössä verkkolaskuformaatit sisältävät paljon enemmän elementtejä.

Finvoice

Finvoice on Suomen Pankkiyhdistyksen ylläpitämä verkkolaskuformaatti (Suomen pankkiyhdistys 2005a). Finvoice on samalla myös verkkolaskujen välityksen mahdollistava välityspalvelu. Finvoice on luotu suomalaisten pankkien lähtökohdista ja se sisältää elementtejä mm. automaattista maksamista varten. Finvoice – laskussa käytetään ebXML:n mukaisia otsikkotietoja sekä SOAP – standardin mukaista kehystä. Sanoma rakentuu kahdesta osasta, itse verkkolaskusta sekä otsikkotiedoista. Otsikkotiedoissa on tiedot laskun reitittämiseen sekä muuta yleistä tietoa ja verkkolasku on erikseen toisessa osassa (kuva 2).



Kuva 2: Finvoice - verkkolaskusanoman rakenne (Suomen Pankkiyhdistys b 2005)

Finvoice – verkkolaskuun kuuluu lisäksi XML Schema - ja DTD – rakennemääritykset, XSL – tyylitiedosto laskun esittämiseksi tietokoneen näytöllä sekä mallilaskut.

eInvoice

eInvoice on pohjoismaisen verkkolaskukonsortion luoma ja nykyisin Elma Oyj Electronic Trading:n ylläpitämä verkkolaskuformaatti. Pohjoismaisen verkkolaskukonsortion jäseniä ovat suuret suomalaiset operaattorit ja pankit. eInvoice on yksi ensimmäisistä suomalaisista verkkolaskuformaateista ja on siksi usein käytössä operaattorien välisessä liikenteessä. eInvoicen ikä näkyy siinä, että eInvoice:sta ei ole julkisesti saatavilla XML-pohjaista

rakennemääritystä eikä mallilaskua. eInvoice on alun perin ollut merkkipohjainen ja siitä on myöhemmin tehty XML-pohjaiset versiot.

TEAPPSXML

TEAPPSXML on TietoEnator Oyj:n luoma ja ylläpitämä verkkolaskuformaatti (TietoEnator 2005). Sitä käytetään pääasiassa Tietoenatorin laskutus- ja kirjanpito-ohjelmistoissa ja näiden ohjelmistojen laajan käytön johdosta TEAPPSXML on yksi kolmesta suuresta verkkolaskuformaatista.

Alla on yhteenveto eri verkkolaskuformaattien sisältämistä ominaisuuksista:

	Finvoice	eInvoice	TEAPPSXML
Uusin versio	1.2 (5.1.2005)	1.4 (22.1.2004)	2.6 (8.11.2004)
Soveltamisohje	On	On	On
DTD	On (424 riviä)	On (157 riviä)	On (561 riviä)
XML Schema	On (1235 riviä)	Ei ole	On (2950 riviä)
XSL	On	Ei ole	Ei ole

Taulukko 2: Verkkolaskuformaattien vertailu

XML Schema - ja DTD–tiedostojen rivimäärä ei korreloi suoraan niiden ilmaisuvoiman kanssa, mutta rivimäärä toimii suuntaa-antavana mittarina. Sen mukaan TEAPPSXML on formaateista laajin ja eInvoice suppein.

Kaikista kolmesta formaatista on olemassa useita versioita ja jokaista formaattia kehitetään jatkuvasti. Tämä tuo haasteita formaattien välisille muunnoksille, joita tarvitaan, kun laskun välityksessä osapuolet (myyjä, ostaja tai operaattori) eivät käytä samaa verkkolaskuformaattia.

3.2.4 Verkkolaskufoorumi

Verkkolaskufoorumi on vuonna 2002 perustettu yhteistyöverkosto jonka tarkoitus on edistää verkkolaskutusta Suomessa. Verkkolaskufoorumi kokoaa yhteen alan keskeiset toimijat kerätäkseen tietoa ja kehittääkseen verkkolaskutusta. Verkkolaskufoorumiin kuuluu pankkeja, operaattoreita, verkkolaskuohjelmistoja tarjoavia ohjelmistotaloja, verkkolaskujen käyttäjäyrityksiä sekä viranomaistahoja (TIEKE 2005b).

Toteuttaakseen tehtäväänsä verkkolaskutuksen edistämisestä Verkkolaskufoorumi julkaisee suosituksia, oppaita ja työkaluja. Työkaluista käytetyin on verkkolaskuosoitteisto, johon on kerätty kaikkien niiden suomalaisten yritysten verkkolaskutiedot, jotka lähettävät tai vastaanottavat verkkolaskuja. Lisäksi Verkkolaskufoorumin toimesta on määritelty kolme tasoa verkkolaskuformaattien tietosisällölle: TIEKE1, TIEKE2 ja TIEKE3.. Tasomäärittelyjen on tarkoitus selkeyttää tilannetta, joka aiheutuu useamman verkkolaskuformaatin käytöstä.

4 XML ja validointi

Tässä luvussa käsitellään tutkimuksen kannalta XML-teknologiaperheen olennaiset osat ja XML:n validointi DTD:n, W3C XML Schema:n ja Schematron:n avulla.

4.1 Yleistä

Extensible Markup Language (XML) on yleinen tiedonkuvauksen metakieli, joka on kehitetty 1990-luvun lopulla toimimaan helppokäyttöisenä ja monipuolisena kielenä (W3C 2004). XML:n suunnittelua ohjasivat toisaalta web-sivujen luomiseen tarkoitettu HTML-kieli sekä toisaalta kansainvälinen dokumenttien kuvaamiseen tarkoitettu metakieli SGML. Näistä XML on sellaisenaan sekä ihmisten että koneiden luettavissa.

XML:n käyttö internetsovelluksissa on lisääntynyt voimakkaasti, mutta tämä on ollut erityisen voimakasta sähköisen liiketoiminnan sovelluksissa. XML:n edut esimerkiksi EDI:in verrattuna ovat potentiaalisesti yksinkertaisemmat ja halvemmat sovellukset sekä parempi luettavuus.

Alla esitellään testauspalvelun kannalta oleelliset osat XML-perhettä: XPath ja XSLT.

XPath

XPath on yksi Extensible Stylesheet Language (XSL) – standardin kolmesta osasta. XPath:ia käytetään viittaamaan XML-dokumentin eri osiin. XPath mallintaa XML-dokumentin sen puumaisen rakenteen kautta ja näin muodostuneen puun osiin voidaan viitata käyttäen XPathia. XPath:in yleisin käyttö on kyselyiden tekeminen XML-dokumenteista. XPath:n syntaksi poikkeaa XML-syntaksista. Tavoite on kompaktiuden ja yksinkertaisuuden saavuttaminen.

Myöhemmin esiteltävä testauspalvelussa käytettävä Schematron – rakennekuvauskieli käyttää XPath - lausekkeita sääntöjen muodostamiseen.

XSLT

XSLT on toinen Extensible Stylesheet Language (XSL) – standardin kolmesta osasta. XSLT on kieli, jonka avulla voidaan muuntaa XML-dokumentti toiseksi XML - dokumentiksi XSLT:n sääntömääritysten mukaisesti. XSLT - dokumentit ovat itse XML – dokumentteja.

Myöhemmin esiteltävä testauspalvelussa käytettävä Schematron – rakennekuvauskielen prosessointi käyttää XSLT – dokumentteja.

4.2 XML:n validointi

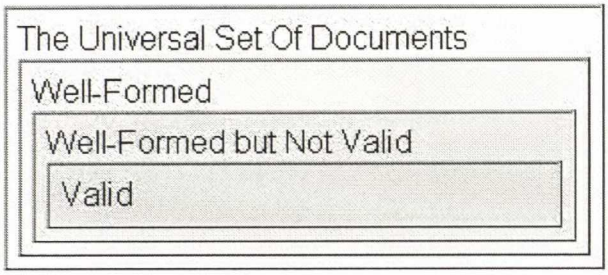
XML:n hyödyntämiseksi koneiden väliseen automaattiseen tiedonvaihtoon täytyy XML-dokumenttien olla jonkun ennalta määritetyn rakenteen mukaisia. Yksinkertaisena esimerkkinä käy päivämäärän esittäminen XML:n avulla. Tämän voi tehdä ainakin seuraavilla tavoilla:

```
<date>
        <year>2005</year>
        <month>8</month>
        <day>1</day>
</date>

<date year="2005" month="8" day="1">

<date>2005-08-01</date>
```

Kuva 3: Kolme tapaa esittää päivämäärä käyttäen XML:ää

Ihminen ymmärtää, että nämä kolme määritystä esittävät samaa päivämäärää. Yksinkertaisin tapa saada tietokoneet tulkitsemaan sama tieto samalla tavalla on kuvata välitettävien XML-dokumenttien rakenne yksiselitteisesti. Kaksi yleisintä rakennekuvaustyyppiä XML-dokumenteille ovat Document Type Definition (DTD) sekä XML Schema, jotka perustuvat kuten XML World Wide Web Consortium:n (W3C) määrityksiin.

XML – dokumentti on *validi* (valid) mikäli sille on olemassa rakennekuvaus ja dokumentti on tämän rakennekuvauksen mukainen. Lisäksi XML – dokumentin täytyy olla hyvin muodostettu (well-formed) (W3C 2004). Hyvin muodostettu XML-dokumentti on XML-syntaksin mukainen. Alla on kuvattu validien ja hyvin muodostettujen XML-dokumenttien keskinäinen järjestys (kuva 4):



Kuva 4: Erityyppisten XML-dokumenttien keskinäinen järjestys

Tässä tutkimuksessa *validointi* tarkoittaa XML – dokumentin tarkastamista rakennemääritystä tai muita sääntöjä ja rajoituksia vastaan.

Alla on esitelty eri tavat validoida XML-dokumentteja (Lim & Wen 2002) (Warren, Reakes & Massari 2003) (Riggs 2003):

DTD ja XML Schema

Yksinkertaisin ja käytetyin tapa validoida XML – dokumentteja. XML Schema on monipuolisempi ja laajempi kuin vanhempaan määritykseen perustuva DTD (Document Type Definition). DTD ja XML Schema on esitelty tarkemmin alla.

Muut kielet rakennekuvausten määrittelyyn

Monet tahot ovat kehittäneet omia rakennekuvauskieliä XML-dokumenttien validointiin. Tunnetuimmat näistä ovat Schematron ja Relax NG, jotka kumpikin ratkaisevat joitain XML Scheman ongelmia. Nämä kielet on esitelty tarkemmin alla.

XSLT

XSLT sisältää monipuoliset säännöt XML-dokumenttien muunnoksia varten ja näitä samoja sääntöjä voidaan käyttää validoimaan XML - dokumentteja. XSLT:n etuna on sen XML-pohjaisuus ja haittana vaikeasti opittava syntaksi.

Ohjelmointikielet

Ohjelmointikielen käyttö XML:n validoinnissa mahdollistaa täysin vapaasti määriteltävät rajoitteet. Haittana on kuitenkin hankala ylläpito, sillä rajoitteiden muuttuessa täytyy ohjelmakoodia muuttaa.

Valmisohjelmat

Markkinoilla on valmiita ohjelmia, jotka kykenevät monimutkaisiin tiedon laadun tarkastuksiin, joista XML:n validointi on yksi osa.

Alla on tarkasteltu yleisimpiä kieliä rakennekuvausten määrittelyyn.

4.2.1 DTD

DTD oli ensimmäinen XML dokumenttien validointiin tarkoitettu kieli. DTD:n etuja ovat sen yksinkertaisuus ja laaja käyttäjäpohja, jonka ansioista DTD:sta on saatavilla paljon tietoa ja työkaluja. DTD:n suurimmat haitat ovat tietotyyppien ja nimiavaruuksien puute sekä DTD:n syntaksi, joka ei ole XML:n mukainen.

Tässä tutkimuksessa DTD tulee esille verkkolaskuformaattien rakennekuvauksissa, jolloin testauspalvelun tulee pystyä käyttämään DTD:n mukaisia kuvauksia.

4.2.2 XML Schema

XML Schema kehitettiin vastaamaan DTD:n heikkouksiin eli XML Schemassa on tuki tietotyypeille ja nimiavaruuksille. XML Schema on itsessään XML – dokumentti, joten sen prosessointiin voidaan käyttää samoja ohjelmia kuin muidenkin XML – dokumenttien prosessointiin.

XML Schema:n heikkouksia ovat kenttien ristikkäisen vertailun puuttuminen sekä tietotyyppien validoinnin rajoitukset (Warren, Reakes & Massari 2003). Kenttien ristikkäinen vertailu tarkoittaa verkkolaskun tapauksessa esimerkiksi verosumman tarkastamista laskemalla se loppusummasta ja veroprosentista.

Tässä tutkimuksessa XML Schema:a käytetään verkkolaskuformaattien rakennekuvauksissa.

4.2.3 Schematron

Schematron on Rick Jelliffen luoma kieli XML – dokumenttien rakennekuvausten määrittelyyn (Schematron 2005). Useimmista muista rakennekuvausten määrittelykielistä poiketen Schematron perustuu *kaavoihin* (pattern) eikä kielioppeihin. Kaavojen avulla voidaan kuvata rajoitteita, jotka eivät ole mahdollisia kielioppipohjaisilla määrittelykielillä kuten XML Schema tai Relax NG.

Schematron käyttää "avoimen validoinnin" periaatetta jossa kaikki on sallittua, mikäli se ei ole erikseen kiellettyä. XML Schema ja muut kielioppipohjaiset rakennekuvauskielet toimivat päinvastoin eli kaikki on lähtökohtaisesti kiellettyä, mikäli sitä ei erikseen sallita. Schematron ja XML Schema ovat kuitenkin toisiaan täydentäviä eikä Schematronia olekaan suunniteltu korvaamaan täysin muita rakennekuvauskieliä.

Schematronin toiminta perustuu kahteen vaiheeseen:

1. Paikannetaan validoinnin kohteena oleva osa XML – dokumentista käyttäen XPath – lausekkeita.
2. Tarkistetaan päteekö toisella XPath – lausekkeella annettu ehto paikannetulle kohteelle.

Schematron käyttää XPath:ia validoinnin kohteen paikantamiseen ja ehdon määrittämiseen. Alla on yksinkertainen XML – dokumentti ja sille schematron – säännöstö, joka tarkistaa löytyykö XML – dokumentista elementti "omistaja" elementin "auto" alta:

```
<auto>
        <merkki>Volvo</merkki>
        <malli>S 80</malli>
        <omistaja>Sven Svensson</omistaja>
</auto>

<sch:schema xmlns:sch="http://www.ascc.net/xml/schematron">
        <rule context="auto">
                <assert test="not(omistaja)">Tällä autolla ei ole omistajaa!</assert>
        </rule>
</sch:schema>
```

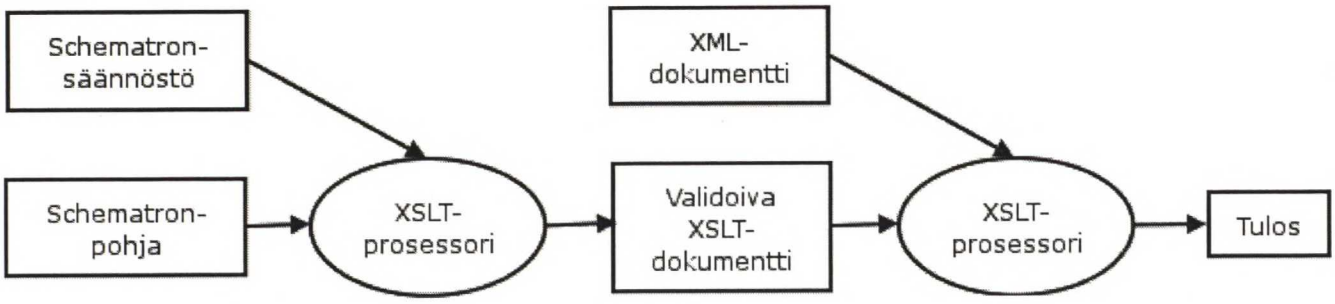
Kuva 5: Esimerkki Schematron - säännöstön käytöstä

Schematron – validointi tapahtuu seuraavan prosessin mukaisesti. Prosessissa on kolme syötettä:

- Schematron - kielinen säännöstö. Tässä on kuvattu testi Schematron - sääntöjen avulla.
- Schematron - pohja. Muuntava XSL-muotoinen dokumentti, joka on osa Schematron – määritystä ja vakio kaikille testeille
- XML-dokumentti. Testauksen kohteena oleva dokumentti.

Prosessi etenee seuraavasti (kuva 6):

1. XSLT – prosessori käyttää Schematron – pohjaa muuntamaan Schematron - säännöstön validoivaksi XSLT - dokumentiksi.
2. XSLT – prosessori muuntaa kohdedokumentin käyttäen juuri luotua validoivaa XSLT – dokumenttia
3. Jälkimmäisen muunnoksen tuloksena syntyy XML-dokumentti, joka sisältää mahdolliset virheilmoitukset.



Kuva 6: Schematron - validoinnin prosessi

Schematronin etuja ovat sen yksinkertaisuus ja pohjautuminen XPath:iin sekä XSLT:een. Näin ollen Schematronin käytön aloittaminen on nopeaa ja edullista eikä vaadi paljon uutta osaamista. Schematronilla on toisaalta muutamia heikkouksia. Jos Schematronia käytetään yhdessä XML Schema:n kanssa, niin validointi jakaantuu kahteen erilliseen dokumenttiin, joita kumpaakin täytyy päivittää. Lisäksi Schematronissa voi olla vain yksi XPath – lauseke yhtä sääntöä kohti, ja tämä ei ole riittävä kaikissa tilanteissa. Heikkoudeksi voidaan laskea myös se, että Schematron ei perustu W3C:n standardiin kuten DTD ja XML Schema. Sen sijaan Schematron on vahvasti ehdolla ISO – standardiksi, joka toisi sille lisää uskottavuutta (Schematron 2005) (Lim & Wen 2002) (Warren, Reakes & Massari 2003).

Schematronia käytetään verkkolaskun testauspalvelussa testien kuvaamiseen. Schematron valittiin muiden vaihtoehtojen sijaan seuraavista syistä:

- Testauspalvelun teknisessä demossa kokeiltiin Schematronia ja se havaittiin tarpeeksi yksinkertaiseksi tähän tilanteeseen.
- Schematronia käytetään myös National Institute of Standards and Technology:n (NIST) luomassa XML – testauspalvelussa. Tästä palvelusta on lisää alla.
- Schematronille löytyi useita valmiita ja käyttökelpoisia toteutuksia sekä kehitystyökaluja (Schematron Java API, Perl XML::Schematron, CS – Wizard)
- Kirjallisuudesta löytyy hyviä kokemuksia Schematronin käytöstä XML:n validoinnissa (Lee & Chu 2002) (Wallentine & Zhou 2002).

5 Testaus

Tässä luvussa käsitellään sähköisen liiketoiminnan testausta sekä testaukseen liittyviä käsitteitä ja termejä. Luvussa käydään myös läpi kaksi olemassa olevaa testauspalvelua referensseinä.

5.1 Testaus yleisesti

Sähköisen liiketoiminnan testaus ja sen testausmenetelmät liittyvät yleisesti ohjelmistotestaukseen. *Ohjelmistotestaus* on keskeinen osa ohjelmistotuotantoa. Perinteisesti testaus on ajateltu ohjelmiston suorittamisena tarkoituksena etsitä siitä virheitä (Myers 1979) Nykyään testaus kytketään asiakasvaatimuksiin eli testauksen tarkoituksena on saada ohjelmisto vastaamaan asiakkaan toiveita (Burnstein 2003).

Ohjelmistotestaus jakaantuu *toiminnalliseen* ja *rakenteelliseen* testaukseen. Toiminnallisessa testauksessa ohjelmisto suoritetaan ja suorituksen tuloksia verrataan ennalta määrättyihin kriteereihin. Rakenteellisessa testauksessa tarkastellaan ohjelmiston lähdekoodia suorittamatta lainkaan ohjelmistoa. Muita keskeisiä ohjelmistotestauksen käsitteitä ovat *testitapaus* (test case), *testijoukko* (test suite), *testipeti* (test bed) sekä *regressiotestaus* (regression testing). Testitapaus on testi, joka kohdistetaan yhteen kohteeseen ja jonka tuloksista voidaan määritellä, läpäiskö kohde testin vai ei. Testijoukko on joukko testitapauksia, jotka liittyvät toisiinsa. Testipeti on ympäristö, jossa testi suoritetaan ja josta osa voi olla luotu vain testin suorittamista varten. Regressiotestauksella tarkoitetaan tietyn testijoukon suorittamista uudestaan kun testauksen kohteeseen on tehty muutoksia. Regressiotestauksen tarkoitus on varmistaa, että tietyt testit läpäistään aina mahdollisista muutoksista huolimatta.

Toiminnallisen testauksen yksi osa-alue on *protokollatestaus*. Protokollatestauksessa testataan protokollien yhdenmukaisuutta, eli vastaako tietyn protokollan toteutus sille annettuja määrityksiä. Protokollatestauksen voidaan katsoa olevan sähköisen liiketoiminnan testauksen esiaste, sillä kummassakin on keskeistä yhdenmukaisuus ennalta annettuja määrityksiä vastaan. Muita yhtäläisyyksiä protokollatestauksen ja sähköisen liiketoiminnan testauksen välillä ovat määritysten esittäminen (formaali tai ei-formaali), standardoidut testijoukot ja testaamisen monimutkaisuus (Bochmann & Petrenko 1994)

5.2 Sähköinen liiketoiminta ja testaus

5.2.1 Yleistä

Erilaisia sähköisen liiketoiminnan viitekehyksiä ja teknologioita on olemassa yhä enemmän ja niitä käytetään yhä laajemmin. Tästä syystä näiden ratkaisujen testaaminen ja oikeellisuuden varmistaminen tulee koko ajan tärkeämmäksi. Sähköisen liiketoiminnan testaus keskittyy suurimmalta osin nimenomaan sähköisten viitekehysten testaukseen. Viitekehyksistä testataan niiden kolmea keskeistä osa-aluetta: prosessia, asiakirjoja ja sanomanvälitystä. Jokaisen osa-alueen testaamiseen on omat, toisistaan eroavat menetelmät ja työkalut.

Sähköisen liiketoiminnan testaus voidaan suorittaa joko matriisitestauksena tai keskitettynä testauksena. Matriisitestaus on yksinkertaisempi järjestää kuin keskitetty testaus, mutta se kasvaa nopeasti mahdottomaksi hallita. Keskitetyn testauksen etuna on pienempi testien määrä, mutta se edellyttää kaikkien toimijoiden keskinäistä yhteistyötä.

Sähköisen liiketoiminnan testauksella varmistetaan ja arvioidaan uusien standardien laatua, edistetään standardien käyttöönottoa käyttäjien ja ohjelmistojen valmistajien keskuudessa, tarkistetaan implementointien yhdenmukaisuus standardia vastaan ja tarkistetaan eri valmistajien tuotteiden välinen yhteentoimivuus (Kulvatunyou et al. 2003). Näistä tärkeimpänä verkkolaskun kannalta on yhdenmukaisuuden tarkistaminen.

Sähköisen liiketoiminnan testauksessa on myös useita haasteita (Durand, Kass & Wenzel 2003). Testauksessa usein käytetty matriisimuotoinen organisointi on kallista, sillä se vaatii runsaasti aikaa sekä työtä. Sähköisen liiketoiminnan erilaiset standardit eivät ole itsessään riittäviä varmistamaan yhdenmukaisuutta. Lisäksi kerran saavutettu yhdenmukaisuus ei säily ilman ylläpitoa, koska organisaatioiden tarpeet, toimintatavat ja järjestelmät muuttuvat jatkuvasti. Yhdenmukaisuus on enemmänkin prosessi kuin tila.

Sähköisen liiketoiminnan testauksen tekee lisäksi vaikeaksi se, että siinä testataan kaikkia sähköisen liiketoiminnan osa-alueita. Esimerkiksi prosessin testaus täytyy tehdä oikeassa ympäristössä oikeilla toimijoilla ja oikealla datalla. Keinotekoisissa testiympäristöissä ei saavuteta luotettavia tuloksia. Yhtenä suurimmista haasteista on myös testauksen kohteena olevan toimijayhteisön sopiminen yhteisistä liiketoiminnallisista vaatimuksista.

Sähköisen liiketoiminnan testauksessa, kuten yleisesti ohjelmistotestauksessa, on hyvä muistaa, että sillä voidaan vain löytää virheitä testauksen kohteesta, ei osoittaa virheiden

puuttumista. Testin läpäissyt sanoma saattaa sisältää virheitä joita ei oltu testissä määritelty tai huomioitu.

Tärkeimpiä työkaluja sähköisen liiketoiminnan testauksessa ovat sähköisen liiketoiminnan testipetit. Testipetejä on toteutettu tämän tutkimuksen alkaessa ainakin ebXML-viitekehykselle.

5.2.2 Käsitteitä

Alla on esitetty sähköisen liiketoiminnan testaukseen liittyvät keskeiset käsitteet.

Syntaksi ja semantiikka

Syntaksi ja *semantiikka* liittyvät kiinteästi toisiinsa ja niitä käsitelläänkin usein yhdessä. Syntaksi on (ohjelmointi)kielen ulkoasua, rakennetta ja kielioppia, kun taas semantiikalla tarkoitetaan (ohjelmointi)kielen merkitystä. Esimerkiksi metakieli XML määrittää syntaksin, mutta ei semantiikkaa. W3C XML Scheman tai Schematron rakennekuvausten avulla voidaan määritellä syntaksi, mutta ei voida ottaa kantaa semantiikkaan. Kuitenkin liiketoiminnallisten dokumenttien validoinnissa semanttinen validointi on keskeistä. Semanttista validointia voidaan tehdä suoraan ohjelmistoon ohjelmoimalla siihen yksityiskohtaisia sääntöjä, mutta tämä tuo mukanaan tunnettuja ongelmia kuten haasteita sääntöjen uudelleenkäytössä ja ylläpidossa sekä sääntöjen sitominen tiettyyn ohjelmointikieleen ja – tapaan (Bussler 2002).

Verkkolaskun testauspalvelussa ei oteta kantaa semantiikkaan vaan ainoastaan syntaksiin.

Yhdenmukaisuus (conformance)

Yhdenmukaisuus tarkoittaa tuotteen, prosessin tai palvelun vastaavuutta määrityksiin (ISO 2004). Yhdenmukaisuutta testaamalla sähköisessä liiketoiminnassa saadaan suurempi todennäköisyys, että tuote on implementoitu oikein määritystä vastaan, on siirrettävissä ja toimii yhteen muiden tuotteiden kanssa (Rosenthal & Brady 2000).

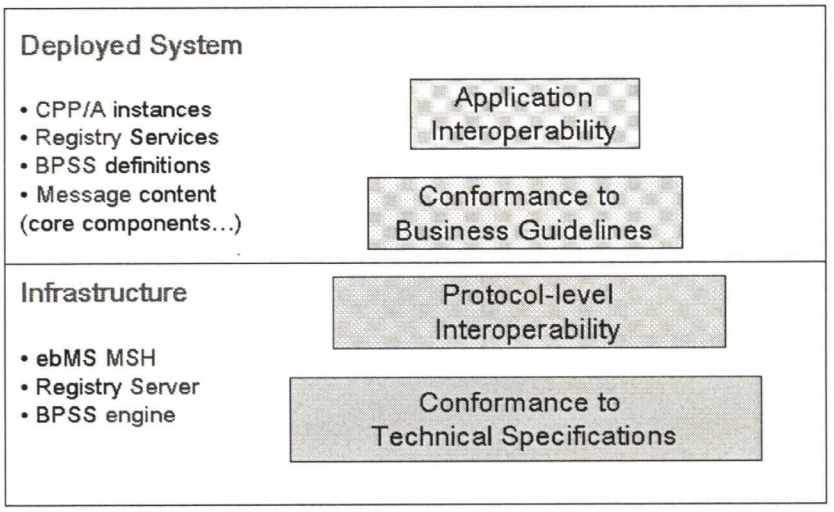
Testauspalvelun kannalta on keskeistä havaita, että yhdenmukaisuus on eri asia kuin yhteentoimivuus. Yhdenmukaisuus on välttämätön, mutta ei riittävä edellytys yhteentoimivuudelle. Testauspalvelussa testataan vain yhdenmukaisuutta ja mahdollinen yhteentoimivuuden testaus otetaan käyttöön myöhemmin.

Yhteentoimivuus (interoperability)

Yhteentoimivuus tarkoittaa kahden tai useamman järjestelmän valmiutta vaihtaa tietoa keskenään siten, että kaikki järjestelmät pystyvät käyttämään vaihdettua tietoa (IEEE 1990).

Yhteentoimivuusongelmat ovat sähköisen liiketoiminnan suurimpia ongelmia ja samalla merkittävimpiä esteitä sähköisen liiketoiminnan laajemmalle käytölle. Yhteentoimivuuden puute aiheuttaa yrityksille ja yhteiskunnalle valtavat kustannukset. Pelkästään Yhdysvaltain autoteollisuus käyttää vuosittain miljardi dollaria järjestelmien välisen yhteentoimivuusongelmien selvittämiseen (NIST 1999). Yhteentoimivuuden testaamisella halutaan ratkaista ongelmat suurimmalta osin ennen kuin järjestelmät on otettu tuotantoon.

Sähköisen liiketoiminnan testauksella pyritään varmistamaan eri toimijoiden välinen *yhteentoimivuus*. Yhteentoimivuus sähköisessä liiketoiminnassa rakentuu kerroksittain ja se muodostaa ns. yhteentoimivuuden kerrosmallin (kuva 7) (Durand, Kass & Wenzel 2003):



Kuva 7: Yhteentoimivuuden kerrosmalli (Durand, Kass & Wenzel 2003)

- Alimassa kerroksessa testataan yhdenmukaisuutta teknisiin määrityksiin. Yhdenmukaisuuden testaus voidaan tehdä jokaiselle toimijalle erikseen ilman muiden toimijoiden läsnäoloa. Verkkolaskun tapauksessa tässä testataan vastaako yksittäinen verkkolasku sille annettua rakennemääritystä ja sovellusohjetta.
- Seuraavassa kerroksessa testataan asiakirjojen sisältävien sanomien välitystä. Tähän testiin tarvitaan kaksi eri toteutusta alemman kerroksen teknisistä määrityksistä. Verkkolaskun tapauksessa tässä testataan sanoman välittymistä lähettäjältä vastaanottajalle oikein.
- Kolmannessa kerroksessa testataan sanomien yhdenmukaisuutta liiketoiminnallisiin määrityksiin. Kerroksessa käsitellään sanomavälityksen yksityiskohtia ja asiakirjojen

sisältämien tietojen semantiikkaa. Verkkolaskun tapauksessa tässä testataan kulkeeko oikein määritelty verkkolasku oikeassa muodossa lähettäjältä vastaanottajalle mahdollisine kuittauksineen ja ymmärretäänkö verkkolaskun sisältö samalla tavalla välitysketjun joka kohdassa.

- Ylimmässä kerroksessa testataan lopullista, sovellusten välistä yhteentoimivuutta. Verkkolaskun tapauksessa tässä testataan verkkolaskun välittymistä suoraan lähettäjän sovelluksesta vastaanottajan sovellukseen siten, että vastaanottajan sovellus tulkitsee verkkolaskun samalla tavalla kuin lähettäjän sovellus.

Yhteentoimivuuden kerrosmallin avulla havainnollistetaan myös testausjärjestys. Mallin alimman kerroksen tulisi olla täysin testattu ennen kuin siirrytään testaamaan seuraavaa ylempää kerrosta.

Sertifiointi

Sertifiointi on osa validointiprosessia. Sertifiointi tarkoittaa jonkun tahon tunnustamien testien läpäisyä onnistuneesti. Sertifiointiin kuuluu neljää osaa (Kim & Yun 2003):

Ensimmäisenä on olemassa määritelmä, joka sisältää ehtoja yhdenmukaisuudesta. Verkkolaskun tapauksessa tätä edustaa verkkolaskuformaattien soveltamisohjeet ja rakennekuvaukset.

Toinen osa on yhdenmukaisuustestien luominen määritelmään pohjautuen.

Kolmas osa on testien suorittaminen ennalta määritetyllä tavalla. Tätä prosessia kutsutaan validoinniksi.

Viimeisenä vaiheena on sertifiointi, jossa testien tulokset tarkastetaan virallisen tahon toimesta ja testauksen kohteelle myönnetään sertifikaatti.

5.3 OASIS IIC ebXML -testausviitekehys

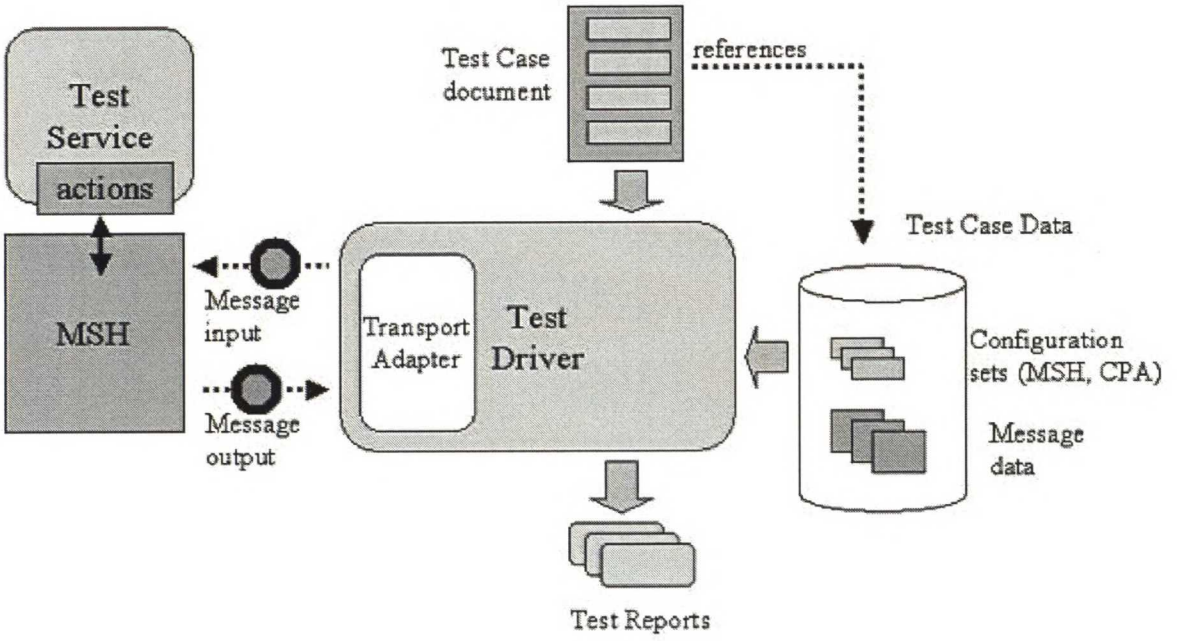
OASIS IIC ebXML testausviitekehys (Testing Framework) on tärkeimpiä ja käytetyimpiä referenssejä sähköisen liiketoiminnan testauksessa. Kyseessä on ebXML:n testaukseen luotu arkkitehtuuri jonka avulla voidaan testata ebXML:n kaikki osa-alueet. Vaikka testausviitekehys on luotu nimenomaan ebXML:ää varten, se on myös itsenäinen kokonaisuus, jonka avulla voidaan periaatteessa testata mitä tahansa XML-pohjaista sähköisen liiketoiminnan viitekehystä.

OASIS IIC ebXML testausviitekehyksellä testataan joko yhdenmukaisuutta tai yhteentoimivuutta. Kummassakin tapauksessa testauksen kohteena on ebXML:n (tai muun viitekehyksen) määritysten mukainen viestinkäsittelijä (Message Handler). Yhteentoimivuustestauksessa kohteena on kaksi viestinkäsittelijää.

OASIS IIC ebXML testausviitekehys määrittelee yleiset testauskomponentit, joista luodaan testausympäristö tiettyä tilannetta varten. Tärkeimmät komponentit ovat testiajuri (Test Driver), testipalvelu (Test Service) ja testitapaus (Test Case). Näistä keskeisin on testiajuri, joka hallitsee ja ohjaa testauksen kulkua. Testiajuri suorittaa testit testitapausten määritysten mukaisesti ja välittää viestejä testauksen kohteelle. Testipalvelu ottaa vastaan viestejä testauksen kohteelta ja vastaa niihin määritysten mukaisesti. Testitapauksissa määritetään mitä testataan ja minkälainen testi ajetaan.

Testejä voidaan joko siten, että testiajuri on palveluna (service) tai, että testiajuri on etäyhteydessä.

OASIS IIC ebXML testausviitekehyksen toimintaperiaate on seuraava:



Kuva 8: OASIS IIC ebXML testausviitekehyksen testiajurin toimintaperiaate (OASIS, 2005)

Testausviitekehyksen mukainen testausprosessi sisältää kuusi vaihetta (OASIS, 2005):

1. Testaussuunnitelman luonti. Luodaan yleinen suunnitelma, jossa määritellään ehdot ja vaatimukset testausta varten
2. Testivaatimusten suunnittelu. Luodaan testivaatimukset määrityksien pohjalta ja kytketään jokainen kohta määrityksistä tiettyyn testivaatimukseen.
3. Testiympäristön suunnittelu. Luodaan testiympäristö testausviitekehyksen komponenteista tiettyä testaussuunnitelmaa varten.
4. Testijoukkojen suunnittelu. Jokaisesta kohdassa kaksi luodusta vaatimuksesta tehdään testitapaus. Testitapaus sisältää alustustiedot, viestiaineiston sekä ehdot testin onnistumisen havaitsemiseksi. Tämän jälkeen testitapaukset ryhmitellään testijoukoiksi. Testijoukot kuvataan XML-dokumentteina.
5. Validointikriteerien luominen. Luodaan kriteerit tiettyä testaustasoa varten. Validointikriteerit määrittävät testaustason sertifioinnin asteen.
6. Testijoukkojen suorittaminen. Suoritetaan testijoukot käyttäen testausviitekehyksen komponentteja.

OASIS IIC ebXML testausviitekehyksen mukaisille toteutuksille on määritelty seuraavat vaatimukset. Näitä vaatimuksia voidaan käyttää tarkistuslistana, kun arvioidaan onko tietty toteutus testausviitekehyksen mukainen:

1. Itsenäinen konfigurointi (self-configuration)
2. ebXML-sanoman muodostaminen
3. Sanoman tallentaminen
4. Tallennetun sanoman käsittely ja haku
5. Testitapauksen suorittamisen hallinta
6. Sanoman lähettäminen
7. Sanoman vastaanottaminen
8. Sanoman sisällön validointi
9. Yhdenmukaisuustestien tulosten raportointi

5.4 NIST QoD – validaattori

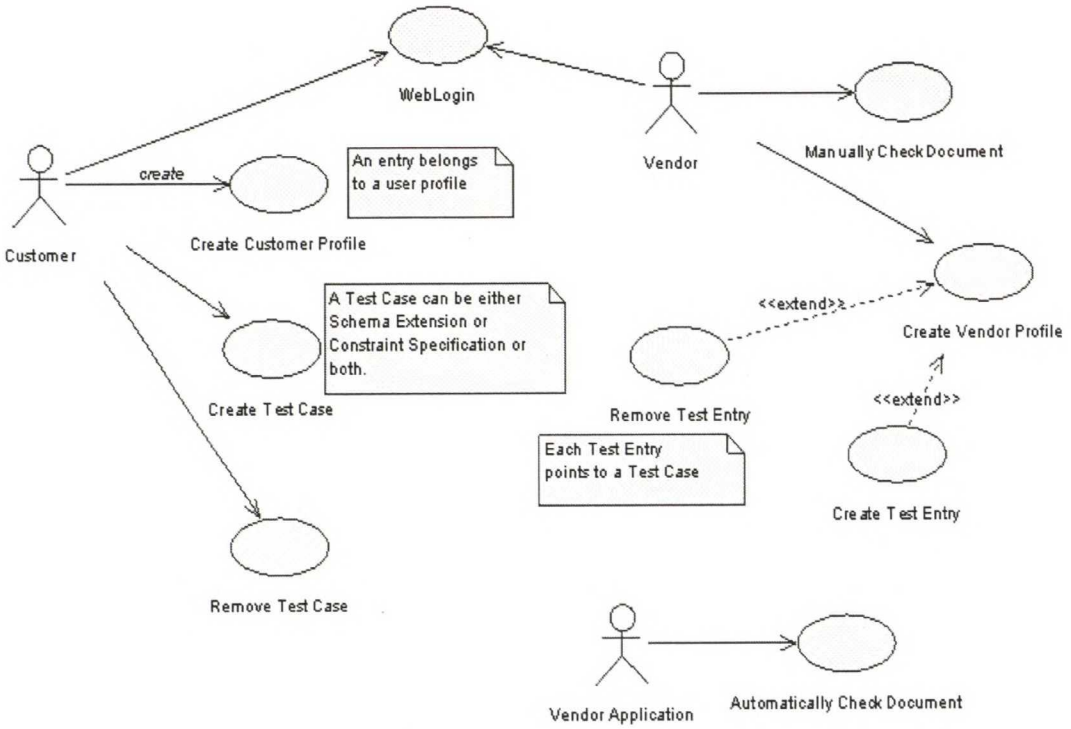
NIST on Yhdysvaltain standardointiorganisaatio. Sen tarkoitus on edistää standardeja ja teknologiaa tuottavuuden nostamiseksi, kaupan edistämiseksi ja elämänlaadun parantamiseksi (NIST 2005a). QoD – validaattori (Quality of Design) on NIST:n kehittämä web-pohjainen XML-muotoisten sanomien testauspalvelu (NIST 2005b). Palvelun avulla voidaan testata sähköisen liiketoiminnan sanomia liiketoiminnallisia sääntöjä vastaan. Liiketoiminnalliset säännöt on määritelty käyttäen Schematronia. QoD-validaattori rakennettiin W3C XML Schema:n testausta varten, mutta sillä voidaan testata mitä tahansa XML-dokumentteja. QoD – validaattori on nostettu esille, koska se oli tämän tutkimuksen alkaessa ainut toimiva ja käytössä oleva Schematron-pohjainen XML-sanomien testauspalvelu josta oli mahdollista saada lisätietoja. Tutkimuksessa saatiin myös käyttöön QoD:n lähdekoodi ja dokumentaatio mikä helpotti validaattorin tarkastelua.

QoD – validaattori on web-pohjainen ja se on suunniteltu useamman käyttäjän palveluksi. Validaattorin toimintaperiaate on ebXML IIC Test Framework:n mukainen. Testauksen valmisteluun kuuluu kaksi vaihetta

- Testivaatimusten luonti.
- Testitapausten luonti. Testitapaukset luodaan testivaatimusten pohjalta

Lisäksi validaattorissa on käsite profiili jonka avulla ryhmitellään tietyn käyttäjän testit yhteen. Profiilin voidaan katsoa vastaavan ebXML IIC Test Framework:n testijoukkoja.

Käyttötapausnäkymä (kuva 9) esittää validaattorin toimintaperiaatteen:



Kuva 9:NIST:n QoD-validaattorin käyttötapauskaavio

Käyttötapauskaaviossa mainitaan vaihtoehto dokumenttien automaattiseen tarkasteluun (Vendor Application -> Automatically Check Document), mutta tätä ei oltu vielä toteutettu. Muuten validaattori toimi kuten esitetty.

5.5 Testauspalveluiden tarjoajia

Markkinoilla on lukuisia kaupallisia sähköisen liiketoiminnan testauspalveluita. Testauspalvelut keskittyvät usein yhden viitekehyksen testaamiseen osa-alueittain tai kokonaisuutena. EbXML-testausta tarjoaa mm. Drake Certivo -niminen yritys ja RosettaNet-testausta suomalainen Mobile-Zoom. Myös erilaiset järjestöt ja organisaatiot ovat rakentaneet testauspalveluita kuten NIST Yhdysvalloissa ja CENORM Euroopassa.

Verkkolaskun testauspalveluita löytyi vain yksi, ruotsalaisen Single Face To Industry (SFTI) -järjestön toteuttama testauspalvelu (SFTI 2005). Palvelu on rakennettu Microsoftin .NET alustalle ja on toimintaperiaatteeltaan samanlainen kuin tässä tutkimuksessa suunniteltu palvelu. SFTI:n testauspalvelun rajoitteena on, että sen validointisäännöt on rakennettu suoraan palvelun lähdekoodiin eikä niitä voi päivittää esimerkiksi rakennekuvaus-tiedostoa

vaihtamalla. Alan nopeaa kehitystä kuvaa hyvin se, että SFTI:n testauspalvelua ei oltu vielä julkaistu tämän tutkimuksen alkaessa.

5.5.1 Verkkolaskun testauspalvelun demo

Osana verkkolaskun testauspalvelun määrittelyä ja suunnittelua rakennettiin palvelusta proof-of-concept -tyyppinen demo. Demon toteutti opiskelijaryhmä Teknillisestä korkeakoulusta keväällä 2005.

Demon alkuperäiset vaatimukset olivat yksinkertaisen XML-validoinnin tekeminen verkkolaskulle ja joidenkin tarkempien virheilmoitusten antaminen. Demo otettiin mukaan osaksi verkkolaskun testauspalvelun vaatimusmäärittelyä jolloin demon alkuperäiset vaatimukset muuttuivat. Uusissa vaatimuksissa painotettiin XML-validoinnin sijaan yksityiskohtaisemman palautteen antamista. Vaatimus XML-validoinnista poistui, koska tälle todettiin olevan tarpeeksi valmiita työkaluja.

Valmistuttuaan demo oli yksinkertainen Schematron-pohjainen validaattori. Demoon oli asetettu muutamia Schematron-sääntöjä joiden avulla pystyttiin testaamaan verkkolaskujen oikeellisuutta. Käyttökokemusten perusteella saatiin seuraavat tulokset:

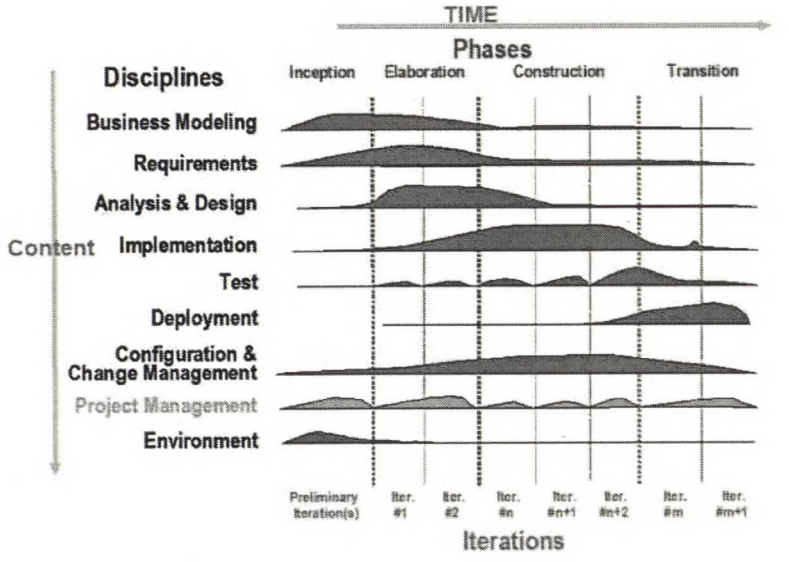
- Schematron on käyttökelpoinen verkkolaskujen validointiin
- Yksinkertainen testauspalvelu voidaan rakentaa pitkälti olemassa olevien komponenttien ja työkalujen päälle
- Suorituskyky ei ole ongelma pienillä testausmäärillä. Yksi validointitapahtuma kesti vain muutamia sekunteja.

6 Menetelmien esittely

Tässä luvussa esitellään ne työtavat ja menetelmät, joilla käytännön osuus tehdään.

6.1 RUP

Tässä tutkimuksessa käytetään RUP (Rational Unified Process)-ohjelmistoprosessimallia koko suunnitteluprosessin osalta. RUP on saavuttanut vahvan aseman yleisesti käytössä olevana ohjelmistoprosessimallina. Se kerää yhteen eri ohjelmistoprosessimallien osia, tukee iteratiivista ohjelmistokehitystä ja edustaa hyviä työtapoja määrittelyssä ja suunnittelussa (Sommerville 2005). RUP valittiin tähän tutkimukseen, koska se on hyvin dokumentoitu ja yleisesti todettu toimivaksi. Alla on esitetty RUP:n mukainen jako vaiheisiin ja osa-alueisiin.



Kuva 10: RUP-prosessimallin vaiheet ja osa-alueet [IBM 2005]

RUP:ta seurataan vain ensimmäisen kahden vaiheen (inception, elaboration) osalta. Muut vaiheet sekä osa-alueet (toteutus, projektinhallinta, muutoshallinta) on rajattu tästä tutkimuksesta pois, vaikka ne ovat osa RUP:ta. Lisäksi osio "business modeling" on jätetty tutkimuksessa vähemmälle huomiolle, sillä testauspalvelun liiketoiminnallinen arviointi ei varsinaisesti kuulunut tutkimukseen.

RUP ei määrittele tarkkaan miten eri osa-alueet (vaatimusmäärittely, suunnittelu, toteutus) tulisi suorittaa. Näin ollen RUP toimii tässä tutkimuksessa ylimmän tason prosessimallina. Vaatimusmäärittelylle sekä arkkitehtuurille valitaan erikseen seurattavat prosessit.

Tutkimus tehdään siten, että ensin laaditaan vaatimusmäärittely ja sen pohjalta arkkitehtuuri ja toiminnalliset määritykset Lopputuloksena halutaan näiden kahden osa-alueen dokumentit

siten, että testauspalvelua voitaisiin lähteä suunnittelemaan ja toteuttamaan näiden dokumenttien pohjalta.

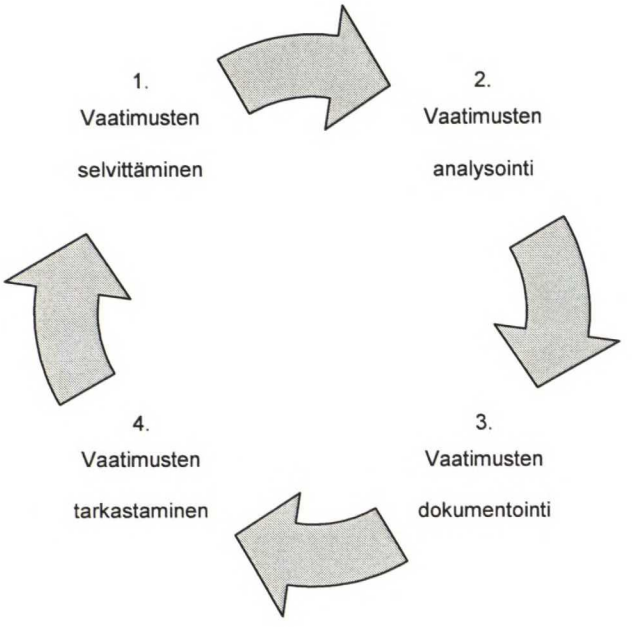
6.2 Vaatimusmäärittely

6.2.1 Yleistä

Vaatimusmäärittelyn rooli ohjelmistotuotannossa ei ole yksiselitteinen. Vaatimusten voidaan nähdä olevan:

- Mitä tahansa joka ohjaa suunnitteluprosessia (Lawrence 1997)
- Kuvaus siitä miten järjestelmän pitäisi toimia (Sommerville & Sawyer 1997)
- Ominaisuus joka tuotteella pitää olla, jotta se tuottaa lisäarvoa sidosryhmälle (Wiegers 2003).

Myös vaatimusmäärittelyprosessista on useita erilaisia näkemyksiä, mutta useimmat niistä ovat luonteeltaan iteratiivisia ja sisältävät seuraavat vaiheet:



Kuva 11: Vaatimusmäärittelyn iteratiivinen prosessi

6.2.2 Prosessi

Tässä tutkimuksessa seurataan Wiegersin (2003) vaatimusmäärittelyprosessia. Wiegers valittiin, koska siinä määriteltiin prosessi selkeästi, siinä oli valmiina kattavat pohjat eri

dokumenteille ja se tarjosi käytännöllisiä neuvoja koko prosessin osalta. Wiegersin mukainen vaatimusmäärittelyprosessi on esitetty alla:

1. Määrittele tavoitetila ja laajuus
2. Tunnista käyttäjäryhmät
3. Tunnista käyttäjät
4. Tunnista päätöksen tekijät
5. Valitse vaatimustenhankinnan menetelmät
6. Tunnista käyttötapaukset
7. Priorisoi käyttötapaukset
8. Luo käyttötapaukset
9. Määrittele laadulliset vaatimukset
10. Dokumentoi toiminnalliset vaatimukset
11. Mallinna vaatimukset
12. Vaatimusten läpikäynti

[Vaiheet 8 - 12 ovat iteratiivisia]

Wiegersin mukaan on olemassa kolmenlaisia vaatimuksia: liiketoiminnallisia vaatimuksia, käyttäjävaatimuksia sekä toiminnallisia vaatimuksia. Tässä tutkimuksessa keskitytään käyttäjävaatimuksiin ja toiminnallisiin vaatimuksiin.

6.2.3 Vaatimusten priorisointi

Vaatimusten priorisointiin on lukuisia erilaisia tapoja. Kuuden erilaisen priorisointimenetelmän vertailussa (Karlsson 1998) lupaavimmaksi nousi analytic hierarchy process (AHP) (Saaty 1980). AHP on yleinen prosessi päätösten tekemiseksi ja soveltuu myös vaatimusten priorisointiin. Siinä priorisointi tehdään ryhmissä vertaamalla aina kahta vaatimusta keskenään. Tässä tutkimuksessa päätettiin kuitenkin käyttää yksinkertaisempaa priorisointimetodia. Syynä oli oletettu tiukka ajankäyttö sidosryhmien kanssa sekä aikaisemman kokemuksen puute priorisoinneista vaatimusmäärittelyn yhteydessä.

Wiegers (1999) esittelee neliportaisen asteikon vaatimusten priorisointiin. Vaatimukset luokitellaan kahden kriteerin, tärkeyden ja kiireellisyyden, mukaan (taulukko 3). Luokittelusta saadaan neljä eri tasoa, joista muokattiin tähän tutkimukseen kolmiportainen asteikko (taulukko 4) jättämällä viimeinen taso (Ei tehdä) kokonaan pois.

	Tärkeä	Ei tärkeä
Kiireellinen	Korkea prioriteetti	Ei tehdä
Ei kiireellinen	Keskitason prioriteetti	Matala prioriteetti

Taulukko 3: Vaatimusten priorisointi kiireellisyyden ja tärkeyden mukaan (Wiegers, 1999)

Priorisointi	Selitys
Täytyy olla (korkea)	Vaatimus täytyy saada ensimmäiseen versioon mukaan
Voi olla (keskitaso)	Vaatimus pyritään saamaan ensimmäiseen versioon, mutta se voidaan myös jättää seuraaviin versioihin. Päätös tehdään toteuttamisvaiheessa, kun toteuttamisen resurssit ovat selvillä.
Jätetään myöhemmäksi (matala)	Vaatimus jätetään suosiolla seuraaviin versioihin.

Taulukko 4: Vaatimusmäärittelyssä käytetty priorisointi

6.3 Arkkitehtuuri

6.3.1 Yleistä

Ohjelmistoarkkitehtuuri ymmärretään usein korkean tason suunnitteluna tai yleiskuvana järjestelmän rakenteesta (Bass, Clements & Kazman 2003). Nämä määritelmät eivät ole kuitenkaan riittävän tarkkoja kuvaamaan arkkitehtuurin roolia ohjelmistokehityksessä. Alla on kaksi määritelmää ohjelmistoarkkitehtuurista:

- "The software architecture of a program or computing system is the structure or structures of the system, which comprises software elements, the externally visible properties of those elements and the relationships among them" (Bass, Clements & Kazman 2003)
- "The selection of structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaboration among those elements" (Kruchten 2004)

Tässä tutkimuksessa käytetään ylempää määritelmää, sillä tutkimuksessa seurataan Bass, Clements & Kazman:n (2003) esittämää Attribute-Driven Design (ADD) -prosessia arkkitehtuurin luomiseen. ADD-prosessi on kuvattu tarkemmin alla.

Arkkitehtuurin tarkastelun yhteydessä esiintyy usein käsitteet arkkitehtuurinen rakenne (architectural structure / architectural viewtype), arkkitehtuurinen tyyli (architectural style / architectural pattern / architectural model) sekä arkkitehtuurinen näkymä (architectural view). Yksiselitteisyyden varmistamiseksi käsitteet on kuvattu lyhyesti alla:

Arkkitehtuurinen rakenne

Ylimmällä tasolla ovat arkkitehtuuriset rakenteet, joita on kolme: rakenne toteutustavan mukaan (module structure), rakenne ajonaikaisten komponenttien mukaan (component-and-connector structure) ja rakenne ohjelmiston ulkopuolisiin komponentteihin liittyen (allocation structure) (Bass, Clements & Kazman 2003) (Clements et al. 2003).

Arkkitehtuurinen tyyli

Arkkitehtuurinen tyyli on rakenteen erikoistapaus. Arkkitehtuurisella tyylillä tarkoitetaan usein toistuvia muotoja (Bass, Clements & Kazman 2003) tai rajoitteita (Clements et al. 2003). Arkkitehtuurinen tyyli on siis kuvaus elementeistä ja niiden välisistä suhteista yhdessä rajoitteiden kanssa. Esimerkkejä arkkitehtuurisista tyyleistä ovat client-server ja kerrosarkkitehtuuri. (Bass, Clements & Kazman 2003) (Clements et al. 2003).

Arkkitehtuurinen näkymä

Arkkitehtuurinen näkymä on arkkitehtuurisen tyylin ilmentymä. Arkkitehtuurinen näkymä on tiettyyn järjestelmään sidottu arkkitehtuurinen tyyli (Clements et al. 2003) tai "leikkaus" tyylistä (Kruchten 2004). Arkkitehtuurinen näkymä on myös jonkun sidosryhmän dokumentoima ja käyttämä kuvaus arkkitehtuurisesta tyylistä (Bass, Clements & Kazman 2003).

Lisäksi Clements et al. (2003) määrittelee käsitteen näkymäryhmä (view set). Näitä ovat mm. Kruchtenin 4+1 – malli (Kruchten 1995) ja Siemens Four View –malli (Soni, Nord & Hofmeister 1995). Kruchtenin 4+1 –malli esitellään tarkemmin alla.

6.3.2 ADD

Attribute-Driven Design (ADD) on Software Engineering Institute:ssa (SEI) kehitetty arkkitehtuurin suunnittelumetodi (Bachmann et al. 2000). ADD – metodissa on keskeistä arkkitehtuurin luominen laadullisten vaatimusten pohjalta. ADD kääntää toisinpäin tunnetun

ohjelmistosuunnittelun periaatteen siitä, että järjestelmän arkkitehtuuri määrittelee pitkälti järjestelmän laadulliset ominaisuudet. ADD:n kaksi keskeistä tavoitetta ovat tukea koko suunnitteluprosessin alkuvaihetta ja mahdollistaa tarkemman suunnittelun aloittamisen aikaisin (Bachmann 2001).

ADD valittiin tähän tutkimukseen, koska se keskittyy arkkitehtuurin suunnittelun alkuvaiheeseen ja koska se sietää hyvin epävarmuutta vaatimuksissa (Bachmann 2001). Lisäksi ADD täydentää ja tukee RUP-prosessia (Bass, Clements & Kazman 2003)

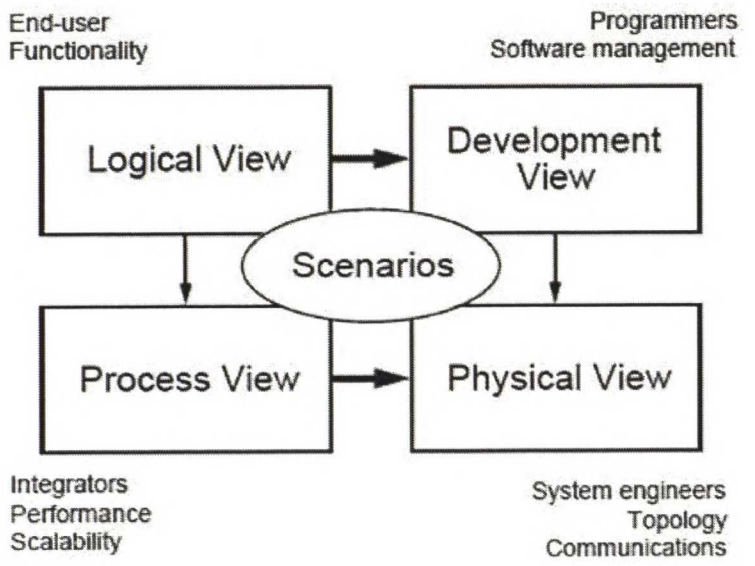
ADD – prosessi toimii seuraavasti: alussa valitaan keskeisimmät arkkitehtuuriset reunaehdot (architectural drivers) joiden pohjalta ADD – prosessi varsinaisesti aloitetaan. Tämän jälkeen ADD:ssa on kolme vaihetta (Bass, Clements & Kazman 2003):

1. Valitse jaettava moduuli. Ensimmäisessä vaiheessa moduuli on koko järjestelmä.
2. Tarkenna moduulia seuraavien ohjeiden mukaisesti
    a. Valitse arkkitehtuuriset reunaehdot laadullisista ja toiminnallisista vaatimuksista.
    b. Valitse arkkitehtuurinen tyyli, joka täyttää nämä valitut vaatimukset.
    c. Luo moduulit ja niiden toiminnallisuus ja esitä tämä käyttäen näkymiä.
    d. Määritä luotujen moduulien rajapinnat.
    e. Varmista, että luodut moduulit ovat yhteneväisiä aikaisemmin luotujen skenaarioiden ja käyttötapausten kanssa.
3. Toista vaiheet 1 ja 2 jokaiselle jaettavalle moduulille.

Prosessin lopussa on olemassa järjestelmän arkkitehtuuri, joka esitetään käyttäen useampia dokumentoituja näkymiä. ADD on luonteeltaan rekursiivinen, joten sen avulla voidaan luoda arkkitehtuuri karkealla tai tarkemmalla tasolla samaa prosessia käyttäen.

6.3.3 Näkymät

Keskeinen osa arkkitehtuuria on sen kuvaamisen käytettävät näkymät. Tässä tutkimuksessa käytetään Kruchtenin (1995) 4+1 – mallin mukaisia näkymiä. Malli valittiin, koska se on keskeinen osa tässä tutkimuksessa käytettävää RUP-prosessia ja se on saavuttanut vahvan aseman arkkitehtuurien kuvaamisessa (Clements et al. 2003). Kruchtenin 4+1 – malli koostuu nimensä mukaisesti neljästä päänäkymästä ja yhdestä täydentävästä näkymästä. Malli on esitetty alla:



Kuva 12: 4+1 -mallin näkymät (Kruchten 1995)

Näkymät kuvataan käyttäen Unified Modeling Language (UML) – kuvauskieltä (Booch, Rumbaugh & Jacobson 1998). UML valittiin, koska se on tiivis osa RUP-prosessia (Kruchten 2004) ja sen käytöstä arkkitehtuurin kuvaukseen on olemassa positiivisia kokemuksia (Medvidoc et al. 2002) (Bass, Clements & Kazman 2003). Näkymät ja niiden suhde UML:n kaavioihin on esitetty alla:

Looginen näkymä (Logical view)

Looginen näkymä esittää järjestelmän toiminnot eli mitä järjestelmä tekee käyttäjille. Näkymässä esitetään tärkeimmät kokonaisuudet ja alajärjestelmät (Kruchten 2004). Näkymä kuvataan käyttäen UML:n luokka- ja oliokaavioita (Booch, Rumbaugh & Jacobson 1998).

Prosessinäkymä (Process view)

Prosessinäkymä esittää järjestelmän ajonaikaiset osat eli tehtävät, säikeet tai prosessit sekä näiden suhteet toisiinsa. Prosessinäkymä keskittyy järjestelmän laadullisiin vaatimuksiin. (Kruchten 2004). Näkymä kuvataan käyttäen UML:n luokka- ja oliokaavioita (Booch, Rumbaugh & Jacobson 1998).

Kehitysnäkymä (Development view)

Kehitysnäkymä esittää järjestelmän lähdekoodin ja komponenttien organisoinnin järjestelmän kehitysympäristössä (Kruchten 2004). Näkymä kuvataan käyttäen UML:n komponentti- ja vuorovaikutuskaavioita (Booch, Rumbaugh & Jacobson 1998).

Sijoittelunäkymä (Physical view)

Sijoittelunäkymä esittää järjestelmän ajonaikaisten komponenttien sijoittelun järjestelmän alustaan tai solmuihin nähden (Kruchten 2004). Näkymä kuvataan käyttäen UML:n sijoittelu- ja vuorovaikutuskaavioita (Booch, Rumbaugh & Jacobson 1998).

Käyttötapaukset (Scenarios)

Käyttötapaukset esittävät järjestelmän toimintaa käyttäjien näkökulmasta. Näkymä ei esitä varsinaisesti järjestelmän rakennetta (Kruchten 2004). Näkymä kuvataan käyttäen UML:n käyttötapauskaavioita (Booch, Rumbaugh & Jacobson 1998).

Tässä tutkimuksessa päätettiin käyttää loogista näkymää testauspalvelun moduulijaon esittämiseen sekä prosessinäkymää keskeisimpien tehtävien esittämiseen. Kehitysnäkymää ei ole käytetty, sillä se ei tässä tapauksessa eroa merkittävästi loogisesta näkymästä. Näiden kahden näkymän ero tulee useimmiten esille vasta isommissa järjestelmissä. Myöskään sijoittelunäkymää ei käytetty, koska koko järjestelmää sijoittuu yhdelle palvelimelle.

Looginen näkymä kuvataan UML:n analyysitason luokkakaaviolla ja prosessinäkymä UML:n sekvenssikaaviolla.

6.4 Vaatimusten tarkastaminen

Vaatimusten tarkastamisella tarkoitetaan vaatimusten oikeellisuuden varmistamista sidosryhmiltä. Tarkastamisen tarkoitus on varmistaa, että vaatimukset ovat oikeita, täydellisiä ja yksikäsitteisiä. Lisäksi varmistetaan, että itse vaatimusmäärittelydokumentti on johdonmukainen, muokattavissa ja jäljitettävissä (Wiegers 2003). Vaatimusten tarkastaminen on perusteltua, koska mahdollisten virheiden korjausten kustannukset nousevat jopa kymmenkertaisiksi ohjelmistoprosessin myöhemmissä vaiheissa (Boehm & Basili 2001).

Vaatimusmäärittelyn tarkastamiseen on erilaisia menetelmiä. Vapaamuotoinen kommentointi (ad hoc review) on vähiten muodollinen tarkastusmenetelmä. Tässä dokumentti annetaan esim. kollegoille luettavaksi ja kommentoitavaksi. Läpikäynnissä (walkthrough) vaatimusmäärittelyn tekijä esittelee dokumentin vertaisryhmälle ja nämä kommentoivat sitä. Varsinainen tarkastus (inspection) on muodollisin tarkastusmenetelmistä. Tarkastuksessa tehtävään koulutettu ryhmä käy dokumentin läpi ennalta määrätyn prosessin mukaisesti.(Wiegers 2003).

Tähän tutkimukseen valittiin tarkastusmenetelmiksi vapaamuotoinen kommentointi ja läpikäynti. Muodollinen tarkastus jätettiin pois, sillä oikean koulutuksen omaavaa henkilöstöä ei ollut saatavilla eikä tutkimuksen aikana ollut resursseja ryhtyä tällaiseen koulutukseen.

7 Nykytilan kartoituksen yhteenveto

Nykytilan kartoituksessa käytiin läpi sähköistä liiketoimintaa ja sähköisen liiketoiminnan viitekehyksiä, verkkolaskutusta ja eri verkkolaskuformaatteja, sähköisen liiketoiminnan testausta sekä vaatimusmäärittelyä ja arkkitehtuurin suunnittelua.

Sähköistä liiketoimintaa käsiteltiin yleisesti ja sähköisen liiketoiminnan viitekehyksien osalta. Kartoituksessa tarkasteltiin XML-teknologioita sekä XML:n validointia. XML:n validointiin on useita erilaisia menetelmiä joista jokaisella on vahvuuksia ja heikkouksia. Validointimenetelmistä valittiin tähän tutkimukseen käytettäväksi Schematron, koska se mahdollisti yksityiskohtaisen palautteen antamisen ollen samalla nopea ja vaivaton päivittää.

Verkkolaskutuksen käsittelyssä tarkasteltiin ensin laskutusta yrityksissä yleensä. Tämän jälkeen siirryttiin verkkolaskutukseen Suomessa, verkkolaskutuksen toimijoihin ja toimintamalleihin sekä verkkolaskuformaatteihin. Suomessa on kaksi keskeistä toimintamallia verkkolaskutuksessa, verkkolaskuoperaattorien yhdysliikennemalli ja pankkien Finvoice-malli. Keskeisiä verkkolaskuformaatteja on kolme: Finvoice, eInvoice ja TEAPPSXML.

Sähköisen liiketoiminnan testaamista käsiteltiin käymällä läpi testaamisen keskeiset käsitteet ja erilaiset testaustavat sekä yksittäisenä esimerkkinä OASIS IIC ebXML testausviitekehyksen toimintaperiaate. Lisäksi tarkasteltiin NIST:n QoD – validaattoria, koska se oli ainut laajemmin käytössä oleva Schematron-pohjainen XML-sanomien testauspalvelu jonka lähdekoodiin ja dokumentointiin oli mahdollista tutustua. Tämän jälkeen esiteltiin lyhyesti opiskelijaryhmän tekemä demo. Demoa käytettiin osoittamaan verkkolaskun testauspalvelun tekniset valinnat oikeiksi.

Lopuksi nykytilan kartoituksessa käsiteltiin vaatimusmäärittelyn ja arkkitehtuurin suunnittelun teoriaa. Vaatimusmäärittelyn yleistä, iteratiivista prosessia tarkennettiin Wiegersin kirjan mukaisesti. Vaatimusmäärittelyyn löydettiin selkeät vaiheet ja vaadittavien dokumenttien rakenne. Arkkitehtuurin suunnitteluun valittiin ADD-menetelmä. Menetelmässä luodaan arkkitehtuuri pitkälti laadullisten vaatimusten pohjalta.

8 Tulokset

Luvussa esitellään verkkolaskun testauspalvelun vaatimusmäärittelyn ja arkkitehtuurisuunnittelun tulokset, toiminnalliset määritykset sekä käydään läpi koko suunnitteluprosessin eteneminen.

8.1 Yleistä

Verkkolaskun testauspalvelun suunnitteluprosessi eteni suunnitelman mukaisesti. Vaatimusmäärittelyn tekeminen aloitettiin nopeasti ja sen tulokset tiivistettiin kahteen dokumenttiin: "Yhteinen tavoitetila" sekä itse vaatimusmäärittely. Vaatimusmäärittelyn tekeminen vei kuitenkin odotettua enemmän aikaa ja arkkitehtuurin suunnittelu alkoi ennakoitua myöhemmin. Vaatimusmäärittelyn arviointia suoritettiin jatkuvasti määrittelyprosessin aikana eikä erillistä, lopullista läpikäyntiä ollut.

Arkkitehtuurisuunnitelma valmistui ajallaan eli neljä kuukautta tutkimuksen aloittamisesta. Vaatimusmäärittelyn odotettua myöhempi valmistuminen aiheutti sen, että arkkitehtuurisuunnitelma aloitettiin myöhässä eikä sen aloittamista voitu nivoa yhteen vaatimusmäärittelyn kanssa. Kiireen takia myös tulosten arviointia ei tehty yhtä paljon kuin oli tavoitteena eikä arkkitehtuurille suoritettu erillistä arviointia.

Toiminnalliset määritykset tehtiin rinnakkain arkkitehtuurin kanssa. Toiminnalliset määritykset kattavat testauspalvelun perustoiminnot ja mahdollistavat palvelun toteutuksen aloittamisen.

Testauspalvelun demo valmistui ajallaan ja täytti sille asetetut vaatimukset.

8.2 Vaatimusmäärittely

Ennen vaatimusmäärittelyn aloittamista määriteltiin ongelma sekä tarkennettiin sidosryhmien kokoonpanoa. Tämän jälkeen yhdessä sidosryhmien kanssa laadittiin vaatimusmäärittely ja arvioitiin vaatimusmäärittelyn onnistumista.

8.2.1 Ongelma

Verkkolaskutuksessa on sekä liiketoiminnallisia että teknisiä ongelmia. Verkkolaskun testauspalvelu keskittyy teknisten ongelmien ratkaisemiseen. Alla käsitellään lyhyesti kumpaakin ongelmaryhmää.

Tekniset ongelmat verkkolaskuissa ovat verkkolaskutiedostojen virheellinen XML sekä verkkolaskuformaatin soveltamisohjeen virheellinen, joko tahallinen tai tahaton, tulkinta. Verkkolaskun välityksen teknisiä ongelmia ovat verkkolaskun välityksen katkeaminen ennen määränpäätä sekä puutteelliset kuittaukset välityksen tilasta. Tekniset ongelmat aiheuttavat epävarmuutta verkkolaskun käyttäjien keskuudessa.

Liiketoiminnalliset ongelmat johtuvat verkkolaskutuksen yleisestä liiketoimintatilanteesta, jossa useat eri toimijat tukevat erilaisia, keskenään osittain ristiriitaisia verkkolaskuformaatteja. Toimijoiden välinen yhteistyö ei ole toteutunut parhaalla mahdollisella tavalla ja sopimus- sekä laskutuskäytännöt ovat käyttäjien näkökulmasta hajanaisia.

Näistä teknisistä ja liiketoiminnallisista ongelmista seuraa, että verkkolaskun käyttö etenee hitaasti, verkkolaskun käyttöönotto on hankalaa ja ongelmatilanteiden selvittäminen on monimutkaista.

8.2.2 Sidosryhmät

Vaatimusmäärittelyn kannalta tärkeimmät sidosryhmät ovat:

- TIEKE Tietoyhteiskunnan kehittämiskeskus ry
- Verkkolaskufoorumin ohjelmistotalotyöryhmän jäsenet
- Verkkolaskuoperaattorit (operaattorit ja pankit)
- Verkkolaskuformaattien haltijat
- Verkkolaskutusohjelmistojen käyttäjät

TIEKE sekä ohjelmistotalotyöryhmä olivat muita sidosryhmiä keskeisemmässä asemassa, sillä vaatimusmäärittely tehtiin hyvin pitkälle näiden kahden sidosryhmän tarpeiden perusteella. Ohjelmistotalotyöryhmän jäsenten tarve oli saada verkkolaskujen tietosisältöä testattua ja siten vähentää ongelmatilanteita. TIEKEn tarve oli yleisesti edistää verkkolaskutusta ongelmatilanteita vähentämällä. Näiden kahden sidosryhmän valinta perustui siihen, että TIEKE on palvelun tilaaja ja ohjelmistotalotyöryhmän jäsenet tulisivat olemaan palvelun ensisijaisia käyttäjiä.

8.2.3 Vaatimusmäärittelyn luominen

Vaatimusmäärittelyn luominen oli voimakkaasti iteratiivinen prosessi ja noudatti pääpiirteiltään nykytilan tarkastelussa esitettyä prosessimallia (ks. 8.2. Vaatimusmäärittely).

Vaatimusmäärittely aloitettiin ongelman tarkastelulla. Tätä varten kerättiin tietoa keskustelemalla TIEKEn asiantuntijoiden kanssa sekä käymällä läpi vanhoja dokumentteja ja muistiinpanoja. Ongelmaa purettiin auki etsimällä tulevan palvelun kaikki sidosryhmät sekä määrittelemällä näiden sidosryhmien alustavia tarpeita.

Tämän jälkeen tarkasteltiin vaatimuksia. Keräysmetodeita olivat haastattelut, sähköpostikyselyt, kirjallisuuden ja vastaavien palveluiden tarkastelu sekä ideointisessiot. Tietoa kerättiin TIEKEn henkilökunnalta, ohjelmistotaloilta, verkkolaskuohjelmistojen asiakkailta ja verkkolaskuoperaattoreilta. Samaan aikaan aloitettiin myös vaatimusten jäljitettävyyden seuranta sekä testauspalvelun demon rakentaminen opiskelijaryhmän toimesta. Demosta on lisää myöhempänä.

Seuraavaksi luotiin dokumentti "Yhteinen tavoitetila" jonka avulla haluttiin varmistaa, että kaikki sidosryhmät ymmärtävät ongelman samalla tavalla. Dokumentissa esitettiin ongelman ratkaisun rajaukset sekä tarkennettiin kohderyhmiä. Tämä dokumentin pohjalta jatkettiin vaatimusten etsimistä.

Vaatimusmäärittelyä käsiteltiin ohjelmistotalotyöryhmän kanssa yhteensä kolme kertaa. Jokaisella kerralla käytiin läpi tähän asti löydetyt vaatimukset, tarkastettiin että ne vastaavat todellisia vaatimuksia sekä ideoitiin yhdessä uusia vaatimuksia. Vastaavan tyyppisten palveluiden ominaisuuksia käytettiin usein ideoinnin pohjana.

Opiskelijaryhmän tekemää demoa käytettiin ideointipalaverien yhteydessä vaatimusten havainnollistamiseksi. Alkuvaiheessa demo sisälsi vain testauspalvelun käyttöliittymän ilman toiminnallisuutta, mutta tästäkin oli hyötyä vaatimusten kohdistamisessa. Myöhemmin demo rakennettiin myös osoittamaan miten laskujen testaaminen tapahtuu käytännössä ja havainnollistamaan Schematron - rakennekuvausten rakennetta.

Yksi viimeisimmistä tiedonkeräysvaiheista oli puhelinhaastattelu testauspalvelun oletetuille loppukäyttäjille. Testauspalvelu oli suunnattu ohjelmistotaloille, joten haastattelun kohteena oli pääasiassa ohjelmistokehityksestä vastaavia henkilöitä. Haastatteluun vastasi kymmenen henkilöä. Haastattelun tuloksena saatiin tarkkoja kuvauksia verkkolaskujen käytännön ongelmista

8.2.4 Ominaisuudet ja toiminnalliset vaatimukset

Vaatimusmäärittelyssä saadut testauspalvelun ominaisuudet on esitetty alla. Jokaisen ominaisuuden kohdalle on listattu siihen liittyvät toiminnalliset vaatimukset sekä näiden vaatimusten tarkemmat selitykset.

8.2.4.1 Tietosisällön testaaminen

Testauspalvelun keskeisin ominaisuus on verkkolaskujen tietosisällön testaaminen. Testaaminen tapahtuu käyttäen verkkolaskuformaattien omia skeemoja sekä erikseen määriteltäviä Schematron - rakennekuvauksia. Verkkolaskun testauspalvelussa päätettiin käyttää rakennekuvauskieliä. Valmisohjelmat todettiin liian raskaiksi verkkolaskun testauspalvelun tarkoitukseen. Testauspalvelun sääntöihin oletetaan tehtävän usein muutoksia, joten ohjelmointikieliin pohjautuva validointi todettiin liian kalliiksi. DTD ja XML Schema eivät ole riittäviä, sillä verkkolaskuformaateista on olemassa jo nämä rakennekuvaukset. XSLT – pohjainen vaihtoehto hylättiin liian monimutkaisena.

Toiminnallinen vaatimus	**Selitys**
Testauspalvelu ilmoittaa verkkolaskun pakollisen kentän tietojen puuttumisesta	Kentän tiedot on merkitty pakolliseksi verkkolaskuformaatin soveltamisohjeessa, mutta laskussa kenttä on tyhjä / puuttuu kokonaan.
Testauspalvelu ilmoittaa virheistä XML-formaatissa	Verkkolasku ei ole validi tai edes hyvin muodostettu.
Testauspalvelu ilmoittaa mikäli verkkolasku poikkeaa kyseisen formaatin W3C Schema tai DTD -määrityksistä	Tämä testaus on helppo tehdä muillakin työkaluilla, mutta se on hyödyllinen lisäominaisuus
Testauspalvelu ilmoittaa verkkolaskun kentän / kenttien väärästä tietosisällöstä	Sovellusohjeen mukaan kentässä pitäisi kulkea tietynlaista tietoa, mutta laskussa tuleekin toisenlaista tietoa
Testauspalvelu ilmoittaa mikäli verkkolasku voidaan muuntaa formaatista toiseen	Tarkastetaan tietyn formaatin mukaisen verkkolaskun muunnoskelpoisuus toiseen formaattiin. Ongelmana on, että formaatit eivät ole suoraan verrannollisia toisiinsa
Testauspalvelu ilmoittaa mikäli verkkolaskun tiedot poikkeavat suosituksesta	Suositukset ovat sääntöjä jotka eivät ole tarpeeksi yleisiä ja/tai hyvin määriteltyjä jotta niitä voisi käyttää suoraan sääntöinä. Suosittelemalla tiettyjä käytäntöjä edistetään yhteisen näkemyksen löytämistä
Käyttäjä/ylläpitäjä voi muokata testisääntöjä	Sääntöjä pitää pystyä lisäämään, poistamaan ja muokkaamaan sitä mukaa kun tarpeet ja/tai verkkolaskuformaatit muuttuvat ja kehittyvät

8.2.4.2 Tasokohtaiset virheilmoitukset

Verkkolaskun tasomäärittelyt auttavat luomaan selkeät

Toiminnallinen vaatimus	**Selitys**
Testauspalveluun voi määritellä laskuformaatti-kohtaiset tasot	Tasot ovat keskeinen osa verkkolaskun testausta. Tasoja tullaan myös muokkaamaan ensimmäisen määrittelyn jälkeen.
Käyttäjä voi testata laskun tasoa vastaan tai avoimena	Käyttäjä haluaa tietää täyttääkö tietty lasku tietyn tason tai kysyä testauspalvelulta minkä tason tietty lasku täyttää.

8.2.4.3 Mallilaskun tilaaminen

Toiminnallinen vaatimus	**Selitys**
Käyttäjä voi tilata testauspalvelusta tietyn formaatin mukaisen mallilaskun	Formaattien mukaiset mallilaskut ovat näin keskitetysti saatavilla. Lisäksi varmistetaan, että kaikki käyttävät samoja mallilaskuja.
Käyttäjä voi tilata testauspalvelusta tietyn säännöstön mukaisen mallilaskun	Laskujen vastaanottoa pitää myös pystyä testaamaan tarkemmilla säännöillä.
Käyttäjä / ylläpitäjä voi lisätä, poistaa ja muokata testauspalvelussa olevia laskuja	Mallilaskuja tulee useita (joka formaatille, joka formaatin versiolle, eri säännöstöille/tasoille). Niiden hallintaan tarvitaan työkalut.

8.2.4.4 Testauksen historiatieto

Toiminnallinen vaatimus	**Selitys**
Käyttäjä voi selata omia / oman yrityksensä aikaisempia testejä	Historiatiedon avulla voidaan välttää turhaa testausta.

8.2.4.5 Laskun tunnistaminen

Toiminnallinen vaatimus	**Selitys**
Testauspalvelu tunnistaa laskun tyypin	Testaus automatisoidaan mahdollisimman pitkälle. Lasku tunnistetaan, mikäli se on laskusta mahdollista tehdä.

8.2.4.6 Laskun näyttö ruudulle tyylisivun avulla

Toiminnallinen vaatimus	**Selitys**
Käyttäjä voi valita laskun kuvan syötettyään laskun testauspalveluun	Testauspalveluun pitää olla syötettynä valmiiksi verkkolaskuformaattien xsl-tiedostot

8.2.4.7 Automaattinen testaus

Toiminnallinen vaatimus	Selitys
Käyttäjä voi lähettää testattavan verkkolaskun koneellisesti testauspalvelun ja saada testin tulokset koneellisesti	Kun testauspalvelulla testataan suuria määriä verkkolaskuja, ei laskujen syöttäminen käsin ole enää käytännöllistä.

8.2.4.8 Reititys

Toiminnallinen vaatimus	Selitys
Käyttäjä voi seurata verkkolaskun kulkua laskutusprosessin aikana	Käyttäjät kokivat ongelmaksi sen, että he eivät saa tarpeeksi tietoa häiriötilanteissa. Tieto siitä missä vaiheessa prosessia häiriöt syntyvät auttaisi häiriöiden korjaamisessa.

8.2.4.9 Sertifiointi

Toiminnallinen vaatimus	Selitys
Testauspalvelu tukee sertifiointia	Sertifiointiprosessia ei määritelty tarkasti, mutta tuki edellyttäisi ainakin mahdollisuutta ryhmitellä erilaisia testejä kokonaisuuksiksi.

8.2.4.10 Käyttäjähallinta

Toiminnallinen vaatimus	Selitys
Testauspalvelu sisältää yksinkertaisen käyttäjähallinnan	Käyttäjäryhmiä olisi alkuun vain kaksi, peruskäyttäjä sekä ylläpitäjä.

8.2.5 Priorisointi

Alla on esitetty testauspalvelun ominaisuuksien priorisointi. Priorisointiasteikkona käytettiin aikaisemmin esiteltyä kolmijakoista asteikkoa.

Ominaisuus	Priorisointi
Tietosisällön testaaminen	Täytyy olla (korkea)
Tasokohtaiset virheilmoitukset	Täytyy olla (korkea)
Mallilaskun tilaaminen	Täytyy olla (korkea)
Testauksen historiatieto	Täytyy olla (korkea)
Laskun tunnistaminen	Voi olla (keskitaso)
Laskun näyttö ruudulle tyylisivun avulla	Voi olla (keskitaso)
Automaattinen testaus	Jätetään myöhemmäksi (matala)
Sertifiointi	Täytyy olla (korkea)
Käyttäjähallinta	Voi olla (keskitaso)

Taulukko 5: Testauspalvelun ominaisuuksien priorisointi

8.2.6 Laadulliset vaatimukset

Alla on esitetty vaatimusmäärittelyssä kerätyt laadulliset vaatimukset.

8.2.6.1 Tietoturva

Tietoturva ei noussut kovinkaan paljon esille haastatteluissa ja ideointipalavereissa. Vaatimuksia olivat käyttäjien tunnistaminen käyttäjätunnus-salasana – yhdistelmällä sekä varmuuskopioiden ottaminen säännöllisesti.

8.2.6.2 Suorituskyky

Suorituskyvyn ainut vaatimus oli, että yhden laskun testaaminen ei saa kestää "kohtuuttoman kauan" tai korkeintaan muutamia sekunteja. Tätä ei pystytty tarkentamaan enempää.

8.2.6.3 Muokattavuus

Muokattavuuden suhteen vaatimukset liittyvät testauspalvelun jatkokehitykseen ja tuleviin lisäominaisuuksiin. Testauspalvelua pitää pystyä laajentamaan myöhemmin mm. automaattisen testauksen mahdollistamiseksi. Lisäksi vaatimusmäärittelyprosessin aikana tuli esiin monia ehdotuksia jotka jätettiin vaatimusmäärittelystä pois. Osa näistä ehdotuksista

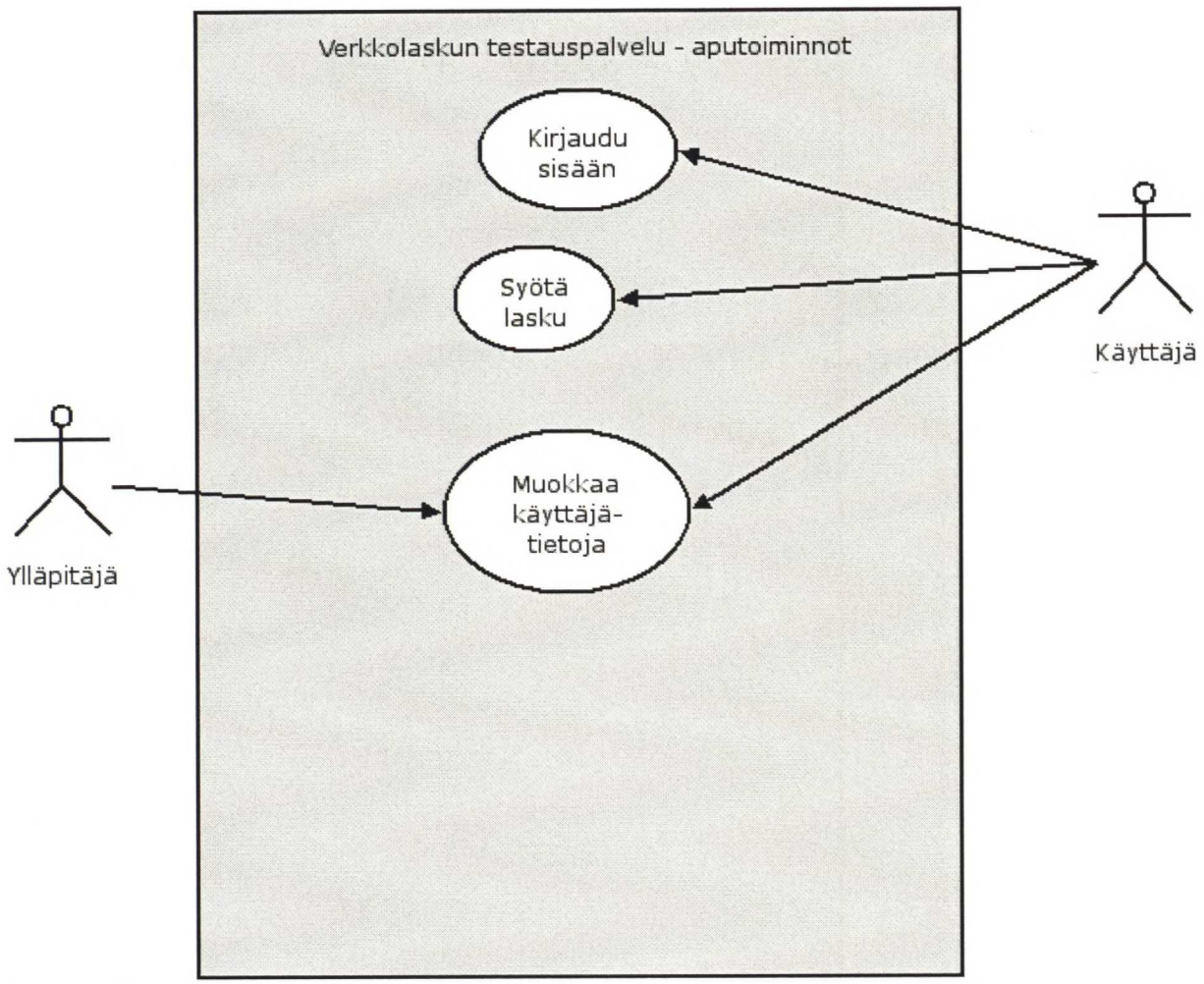
nousee varmasti esille myöhemmin. Vaikka ehdotuksia ei tarkennettu eikä kirjattu ylös on selvää, että testauspalvelua tullaan muokkaamaan jatkossa.

8.2.6.4 Muuta laadulliset vaatimukset

Muista laadullisista vaatimuksista (saavutettavuus, siirrettävyys) ei tullut mitään vaatimuksia. Kyseessä on pohjimmiltaan yksinkertainen web-palvelu ja sidosryhmät eivät kokeneet tarpeelliseksi määritellä palvelua tarkasti laadullisten vaatimusten kautta.

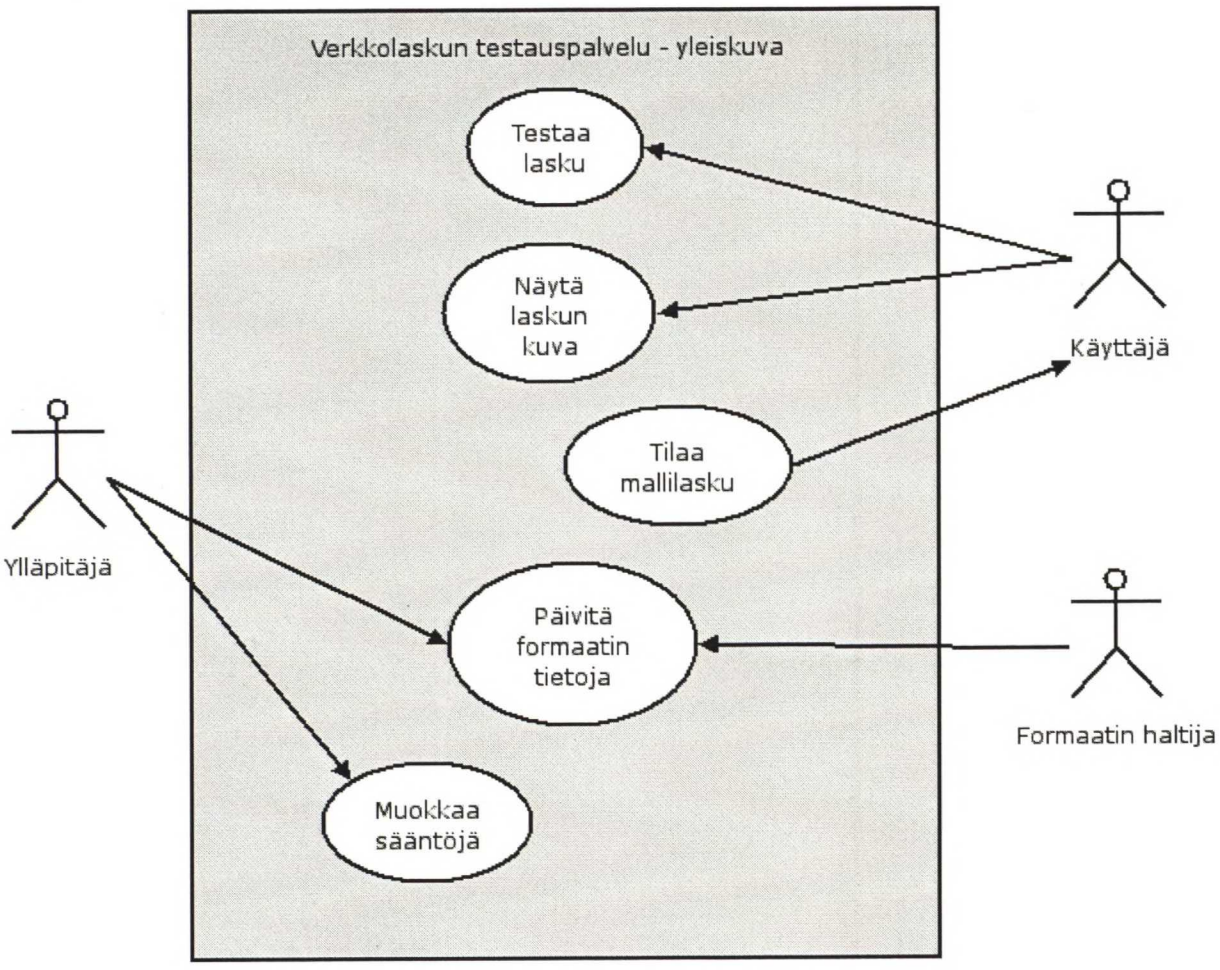
8.2.7 Käyttötapaukset

Kuvassa 13 on esitetty testauspalvelun ensimmäinen käyttötapaus. Tässä käyttötapauksessa on kuvattu ne toiminnot, jotka eivät koske suoraan verkkolaskun testausta.



Kuva 13: Testauspalvelun käyttötapaus - aputoiminnot

Kuvassa 14 on esitetty testauspalvelun toinen käyttötapaus. Tässä käyttötapauksessa on kuvattu suoraan verkkolaskun testaukseen liittyvät toiminnot.



Kuva 14: Testauspalvelun käyttötapaus - yleiskuva

8.2.8 Vaatimusmäärittelyn tarkastus

Vaatimusmäärittelyn tarkastamista suoritettiin jatkuvasti iteratiivisen vaatimusmäärittelyprosessin mukaisesti. Vaatimusmäärittelylle suoritettiin lyhyitä läpikäyntejä säännöllisissä tapaamisissa. Sidosryhmät olivat mukana kaikissa tapaamisissa ja vaatimusmäärittelyä muokattiin tapaamisissa esitettyjen kommenttien mukaisesti.

8.3 Arkkitehtuuri

8.3.1 Arkkitehtuurin luominen

Arkkitehtuurin suunnittelu aloitettiin hieman ennen vaatimusmäärittelyn lopullista valmistumista.

Arkkitehtuuri luotiin vaatimuksista ADD - prosessin mukaisesti.. Arkkitehtuurin suunnittelussa keskeiseksi nousi kolme laadullista vaatimusta, jotka ovat tärkeysjärjestyksessä muokattavuus, testattavuus ja turvallisuus.

Muokattavuus on tärkein vaatimus, koska testauspalvelulla on monta sidosryhmää, joilla on osittain toisistaan poikkeavat tarpeet. Vaatimusmäärittely toteutettiin tietoisesti vain yhden ryhmän (ohjelmistotalotyöryhmän jäsenet) tarpeiden perusteella ja tästä syystä testauspalveluun on odotettavissa muutoksia tulevaisuudessa. Lisäksi vaatimusmäärittelyyn kirjattiin vaatimuksia, jotka haluttiin vasta palvelun myöhemmissä versioissa. Arkkitehtuurin tulee mahdollistaa nämä tulevat muutokset ja joustaa muutostarpeiden mukaan.

Testattavuus on myös tärkeää, koska testauspalvelun täytyy olla uskottava käyttäjien silmissä, jotta palvelua voidaan käyttää osana sertifiointiprosessia. Uskottavuuden saamiseksi testauspalvelua täytyy voida testata kattavasti. Testattavuus ei ole kuitenkaan tärkein vaatimus, koska palvelu on varsin yksinkertainen ja testitapausten luonti ja suorittaminen ei ole erityisen vaikeaa. Lisäksi testitapauksia ei tule paljon, koska palvelun ydin sisältää vähän toimintoja.

Turvallisuus on näistä kolmesta laadullisesta vaatimuksesta vähiten tärkeä, sillä testauspalvelussa testattavat dokumentit ovat eivät ole täysin salaisia (laskutusohjelmistojen käyttäjät pystyvät tarkastelemaan dokumentteja) ja dokumentit pohjautuvat joka tapauksessa täysin julkisiin suosituksiin (esim. Finvoicen soveltamisohje). Testattavissa dokumenteissa saattaa kuitenkin olla oikeaa dataa (esim. tietoa asiakkaan hinnoittelusta), joten palvelu täytyy olla suojattu.

8.3.2 Looginen näkymä ja toiminnalliset määritykset

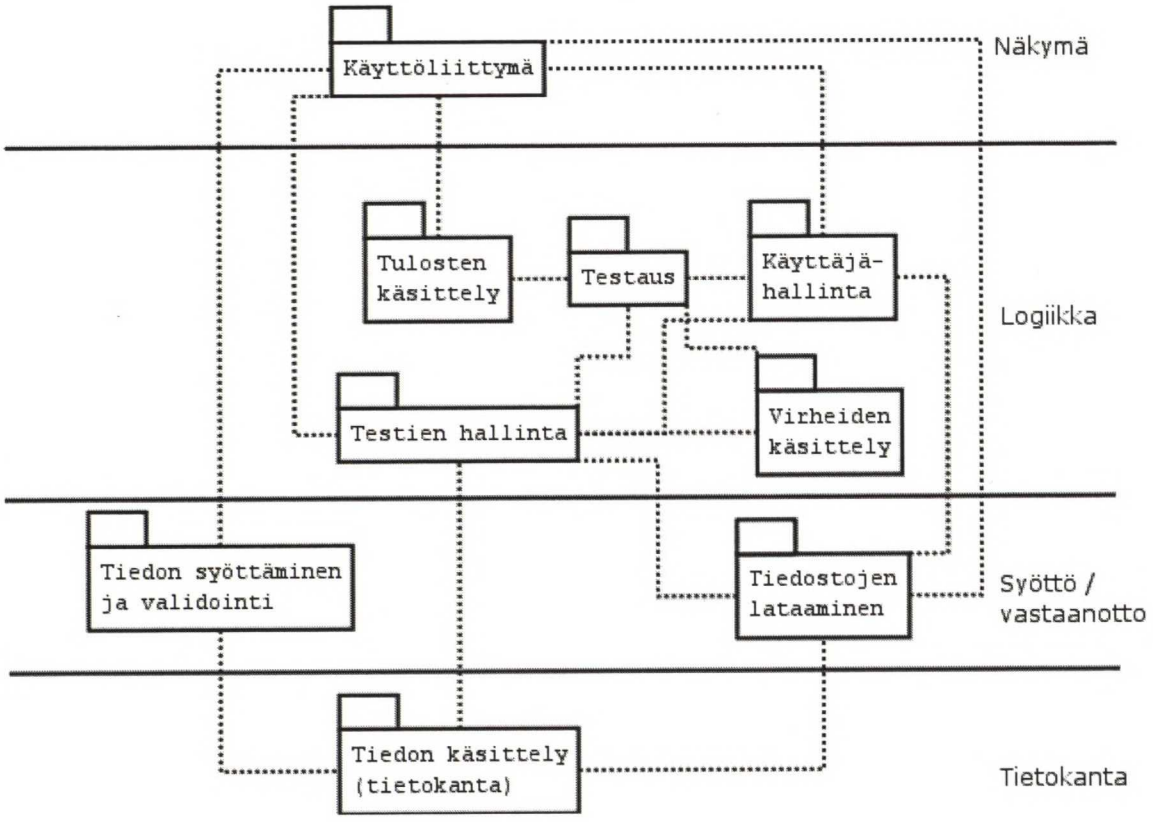
Kaaviossa esitetään testauspalvelun moduulit, niiden väliset rajapinnat sekä moduulien jakautuminen eri tasoihin. Jokaisen moduulin toiminnalliset määritykset on myös listattu.

Moduulit ja niiden toiminnallisuudet kuvataan tarkemmin alla.

Moduulien väliset *rajapinnat* on kuvattu kaaviossa vain viitteellisesti. Rajapintojen kuvaamisen tarkoitus on korostaa testauspalvelun muokattavuutta pitämällä moduulien väliset yhteydet minimissä.

Tasot kuvaavat moduulien jaottelua kerrosarkkitehtuurin mukaisesti. Perusajatuksena on näkymien ja toiminnallisen logiikan erottaminen toisistaan. MVC (Model-View-Controller) – suunnittelumallia ei ole käytetty suoraan, koska testauspalvelu on suhteellisen kevyt ja yksinkertainen palvelu ja MVC-mallin täsmällinen toteuttaminen on työlästä. Lisäksi MVC – mallin suurin hyöty tulee näkymien vaihtuessa (esim. eri päätelaitteille) ja testauspalvelun tapauksessa käyttöliittymä on pienessä roolissa eikä tällaisia uusia päätelaitteita ole

odotettavissa testauspalvelulle. Tiedon syöttäminen ja vastaanottaminen on omassa kerroksessaan, koska siihen on tulossa suurella varmuudella eniten muutoksia lähitulevaisuudessa.

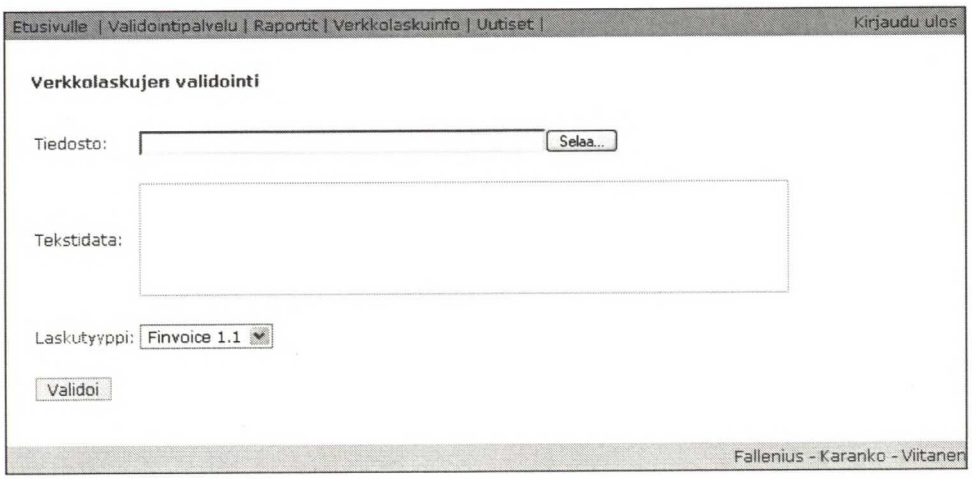


Kuva 15: Testauspalvelun moduulinäkymä

8.3.2.1 Käyttöliittymämoduuli

Käyttöliittymämoduuli vastaa testauspalvelun käyttöliittymästä. Käyttöliittymä vastaa palvelun ulkoasusta eri toimintojen mukaisesti sekä välittää käyttäjän syötteen eteenpäin. Tässä arkkitehtuurissa ei ole noudatettu Model-View-Controller (MVC) – mallia, jolloin erillistä Controller – moduulia ei ole. Tällaisen toteuttaminen on kuitenkin mahdollista jolloin Controller ottaisi vastaan käyttäjän syötteen käyttöliittymämoduulilta.

Yksi esimerkki käyttöliittymästä on alla oleva kuva. Kyseessä on testauspalvelun demo-version käyttöliittymä. Testauksen kohteena oleva lasku voidaan syöttää joko tiedostona tai kopioimalla lasku tekstimuodossa ikkunaan. Lisäksi käyttäjä voi valita mitä verkkolaskuformaattia vastaan hän haluaa testata.



Kuva 16: Esimerkki testauspalvelun käyttöliittymästä

Toiminnot

- Esittää käyttäjälle testauspalvelun eri näkymät (web-sivut)
- Ottaa vastaan tiedon muista moduuleista ja muuntaa sen esitettävään muotoon (HTML)
- Ottaa vastaan käyttäjän syötteen (napit, valikot, syötekentät) ja välittää ne eteenpäin muille moduuleille
- Näyttää käyttäjille palvelun sisäiset virheilmoitukset mikäli sellaisia tulee

8.3.2.2 Tiedon syöttäminen ja validointi -moduuli

Moduuli vastaa kaikista käyttäjän syöttämistä dokumenteista. Näitä dokumentteja ovat verkkolaskuformaattien soveltamisohjeet, rakennekuvaukset, mallilaskut, tyylitiedostot sekä käyttäjien syöttämät testattavat laskut. Moduuli validoi XML-dokumentit ja ilmoittaa käyttäjälle, mikäli dokumentti ei ole validi. Tämä tehdään siksi, ettei tällaisia XML-dokumentteja voi käyttää myöhemmin testeissä ja muut moduulit olettavat XML-dokumentin olevan validi. Käyttäjien pitäisi joka tapauksessa suorittaa XML-validointi kaikille dokumenteille ennen niiden syöttämistä testauspalveluun eli tämä toiminnallisuus on vain varotoimenpide.

Moduuli tunnistaa sille syötetyn laskun tyypin (Finvoice, TEAPPSXML, eInvoice) sekä version mikäli se on mahdollista.

Moduuli myös tallentaa dokumentit välittämällä ne eteenpäin tiedon käsittely -moduulille. XML-muotoiset dokumentit tallennetaan vain jos ne ovat valideja.

Tämä toiminnallisuus on erotettu omaksi moduuliksi, koska testauspalveluun tullaan tulevaisuudessa rakentamaan mahdollisuus syöttää testattavia dokumentteja koneellisesti.

Toiminnot

- Tarkastaa käyttäjän syöttämät dokumentit
    - Mikäli dokumentti on XML muotoinen, moduuli tarkastaa onko se validia XML:ää
    - Syöttötapa voi olla joko tiedoston lataaminen palveluun tai laskun syöttäminen tekstinsyöttökentän kautta
- Tunnistaa laskuformaatin tyypin mikäli se on mahdollista tehdä
    - Finvoicessa ja TEAPPSXML:ssä on erillinen elementti, joka kertoo laskuformaatin ja version
- Välittää dokumentit eteenpäin tietokannalle

8.3.2.3 Tiedon käsittely (tietokanta) -moduuli

Moduuli ottaa vastaan pyyntöjä hakea, muokata, lisätä tai poistaa tietoa. Moduuli ei puutu mitenkään tiedon laatuun eikä jalosta haettua tietoa itse. Nämä tehtävät kuuluu kutsuvien moduulien hoitaa itse. Näin saadaan testauspalvelu mahdollisimman riippumattomaksi sen käsittelemästä tiedosta. Tämä on tarpeellista, koska testauspalvelun käyttö voi laajentua koskemaan erilaista tietoa (esim. muuta kuin XML-muotoista dataa)

Toiminnot

- Vastaa kaiken tiedon (dokumenttien, käyttäjätietojen, testidatan) hallinnoinnista

8.3.2.4 Testien hallinta - moduuli

Moduulin avulla käyttäjä hallitsee testejä. Käyttäjä voi lisätä, poistaa ja muokata testejä sekä selata palvelussa valmiina olevia testejä. Testien tietoihin kuuluu nimi, kuvaus ja säännöstö (Schematron – rakennekuvaus). Testeistä ylläpidetään lokia, jossa on tieto siitä mikä testi on ajettu, milloin se on ajettu, minkä käyttäjän toimesta ja mikä oli testin syöte sekä tulos.

Testien hallinta – moduuli koordinoi myös useampien testien ajamista peräkkäin. Tätä toiminnallisuutta tarvitaan ainakin sertifiointiprosessissa. Testejä voidaan ryhmitellä ja ajaa

näitä ryhmiä yhtenä kokonaisuutena. Käyttäjä voi luoda uusia ryhmiä, muokata ryhmiä lisäämällä ja poistamalla testejä sekä poistaa ryhmiä.

Testien hallinta - moduulissa voi tarkastella myös testauksen kohteena olevaa laskua tai aikaisempien testien laskuja visuaalisessa muodossa, mikäli tälle laskuformaatille on syötetty vaadittavat XSL- tyylitiedostot.

Toiminnot

- Yksittäisten testien hallinnointi
- Testiryhmien hallinnointi
- Laskun visualisointi

8.3.2.5 Testaus

Testauspalvelun ydin on testausmoduuli. Moduuli ottaa vastaan testattavan dokumentin ja säännöstön ja palauttaa testin tuloksen. Schematron - säännöt on rakennettu siten, että palaute on suoraan säännöissä mukana. Mikäli testauksessa käytetään jotain sellaista rakennekuvauskieltä, jossa tarvitaan useampia sääntödokumentteja tai erillistä dokumenttia palautetta varten, täytyy testausmoduulin mahdollistaa myös tämä.

Testausmoduuli tulostaa tarvittaessa yksityiskohtaista palautetta testauksen etenemisestä. XSLT-prosessorissa on syytä olla tällainen ominaisuus valmiina ja testausmoduulin tulee pystyä käsittelemään näin syntynyttä tietoa.

Moduuli on syytä pitää hyvin itsenäisenä kokonaisuutena, sillä siihen voi tulla muutoksia testauspalvelun kehittyessä testaamaan muita sähköisiä sanomia kuin verkkolaskuja. Tällöin voi tulla tarvetta muillekin rakennekuvauskielille kuin Schematron.

Toiminnot

- XML-dokumenttien testaus käyttäen valmiiksi luotuja sääntöjä
- Testitulosten välittäminen eteenpäin tulosten käsittely – moduulille
- Testitulosten välittäminen eteenpäin testin hallinta – moduulille tallentamista varten

8.3.2.6 Tulosten käsittely

Moduuli ottaa vastaan testien tulokset testausmoduulilta ja prosessoi tulokset käyttäjän toiveiden mukaisesti. Lähtökohtaisesti tulokset esitetään käyttäjälle testauspalvelussa kuten muutkin palvelun näkymät. Moduuli on kuitenkin erotettu omaksi toiminnoksi siltä varalta,

että tulevaisuudessa halutaan saada testauksen tulokset automaattisesti lähetettyä käyttäjälle esim. johonkin toiseen testausohjelmistoon. Kun tämä toiminnallisuus yhdistetään tietojen automaattiseen syöttöön, voidaan ajaa hyvin pitkälle automatisoituja testejä.

Tulosten käsittely – moduuli voi ottaa syötteenä myös erilaisia tyylitiedostoja joilla tulokset saadaan haluttuun muotoon.

Toiminnot

- Testitulosten esittäminen

8.3.2.7 Tiedostojen lataaminen

Tämä moduuli esittää tarjolla olevat tiedostot ja antaa käyttäjälle mahdollisuuden ladata tiedostoja. Näitä tiedostoja ovat verkkolaskuformaattien mukaiset mallilaskut sekä formaattien soveltamisohjeet.

Moduuli on erotettu omaksi toiminnoksi, koska ainakin mallilaskuja voidaan haluta tilata jatkossa myös automaattisesti.

Toiminnot

- Hallinnoi ladattavia tiedostoja ja mahdollistaa tiedostojen lataamisen käyttäjälle

8.3.2.8 Käyttäjähallinta

Käyttäjähallintamoduuli vastaa käyttäjien todennuksesta (authentication) ja valtuutuksesta (authorization). Moduuli ottaa vastaan käyttäjän kirjautumistunnukset (käyttäjätunnus ja salasana) ja varmistaa, että nämä vastaavat käyttäjähallintaan tallennettuja tietoja. Käyttäjähallintamoduuli ottaa myös vastaan kyselyitä muilta moduuleilta ja palauttaa tiedon onko käyttäjällä valtuudet jonkun toiminnon suorittamiseen.

Käyttäjiä voidaan lisätä ja poistaa sekä heidän tietoja voidaan muokata. Käyttäjärooleja ovat ylläpitäjä ja peruskäyttäjä. Rooleja on mahdollista lisätä ja poistaa. Rooleja voi myös muokata siten, että roolille annetaan oikeus tietyn moduulin tietyn toiminnon suorittamiseen tai tämä oikeus otetaan pois.

Käyttäjät on jaettu myös käyttäjäryhmiin, jotka ovat yleensä yrityksiä. Näin voidaan luoda samalle yritykselle useita käyttäjiä, jotka voivat käyttää samoja, yhteisiä testitapauksia.

Tietoturvan lisäämiseksi käyttäjähallintaan voidaan tallentaa käyttäjien suorittamat toiminnot ja toiminnon suorittamisaika jolloin saadaan kirjausketju ongelmatilanteiden selvittämiseksi.

Toiminnot

- Käyttäjien hallinta
- Käyttäjäroolien hallinta
- Käyttäjäryhmien hallinta
- Käyttöoikeuksien muokkaaminen moduuli-, ryhmä- ja roolikohtaisesti

8.3.2.9 Virheiden käsittely

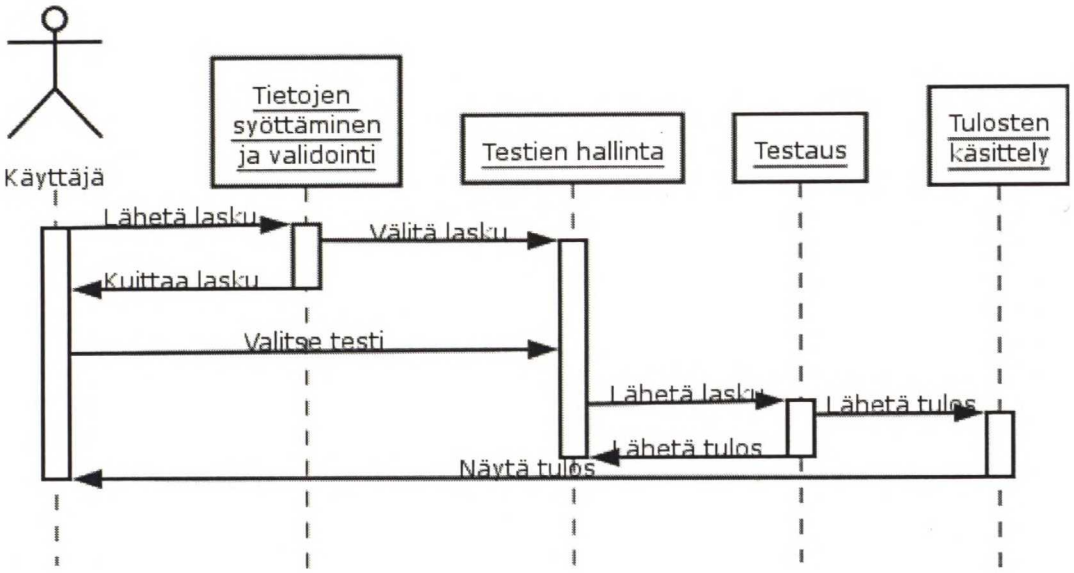
Virheiden käsittely – moduulin tehtävä on hallita keskitetysti sisäisten virheiden käsittelyä koko testauspalvelussa. Kaaviossa rajapinnat on piirretty moduuleihin testaus ja testien hallinta, koska nämä ovat virheiden käsittelyn kannalta tärkeimmät. Tarve tälle moduulille tulee testattavuusvaatimuksesta jonka mukaan palvelua pitää pystyä testaamaan helposti.

Toiminnot

- Ottaa vastaan virheilmoituksia muilta moduuleilta
- Näyttää virheilmoitukset ennalta määritellyllä tavalla käyttöliittymän kautta

8.3.3 Prosessinäkymä

Alla on esitetty testauspalvelun perusprosessi, yksittäisen laskun testaus, sekvenssikaaviona.



Prosessi alkaa käyttäjän toimesta, kun tämä lähettää laskun testauspalveluun. Mikäli lasku läpäisee ensimmäisen validoinnin, se lähetetään eteenpäin testien hallintaan ja käyttäjälle kuitataan laskun onnistunut vastaanotto.

Testien hallinnassa vaaditaan jälleen käyttäjän syötettä testin valinnassa. Valinnan jälkeen lasku lähetetään testattavaksi. Testin tulos lähetetään eteenpäin tulosten käsittelyyn, jossa se esitetään käyttäjälle hänen haluamassa muodossa. Lisäksi testin tulos lähetetään testien hallintaan jossa lasku, ajettu testi ja testin tulos tallennetaan.

8.3.4 Rajapinnat

Testauspalvelun rajapinnat ulospäin toteutetaan käyttäen HTTP ja FTP -protokollia. Verkkolaskun vastaanoton testauksessa HTTP-protokollan yli tulee HTTP-kutsu joka sisältää verkkolaskun. Verkkolaskun lähetyksen testauksessa HTTP-protokollan yli tulee HTTP-vastaus, joka sisältää halutun malliverkkolaskun. Manuaalisessa testauksessa käytetään HTTP-protokollaa, koska testauspalvelun käyttö tapahtuu www-sivujen avulla ja HTTP-protokollan toteuttaminen on yksinkertaista.

Laskun automaattisessa vastaanotossa käytetään FTP-protokollaa. Testauksen kohteena olevat laskut siirretään yhdessä tiedostossa testauspalveluun jossa ne otetaan yksi kerrallaan käsittelyyn. Automaattisessa vastaanotossa käytetään FTP-protokollaa, koska se on tällä hetkellä yleisesti käytössä verkkolaskujen välityksessä. Laskujen välitys on myös usein salattua käyttäen esimerkiksi VPN-tekniikoita.

8.3.5 Arkkitehtuurin tarkastus

Arkkitehtuurisuunnitelma tarkastettiin sidosryhmien toimesta useaan otteeseen. Arkkitehtuuri oli tarkoitus arvioida erityisten arkkitehtuurin arviointimenetelmien mukaan, mutta tähän ei jäänyt aikaa. Arkkitehtuurisuunnitelman tärkein kriteeri oli se, että suunnitelman pohjalta olisi pitänyt voida suoraan aloittaa testauspalvelun toteuttaminen. Toteuttamisvaihe siirtyi kuitenkin eteenpäin aiotusta ajankohdasta ja toteuttaminen ei alkanut tämän tutkimuksen valmistuessa. Toteuttamisesta tarjouksen tehnyt toimittaja käytti kuitenkin arkkitehtuurisuunnitelmaa tarjouksen pohjana. Toimittajan mukaan arkkitehtuurisuunnitelma oli tarpeeksi kattava testauspalvelun toteuttamisen aloittamiseksi.

9 Tulosten tarkastelu

Tässä luvussa tarkastellaan tutkimuksen tuloksia eli vaatimusmäärittelyä, arkkitehtuuria ja toiminnallisia määrityksiä ja arvioidaan näiden tulosten luotettavuutta.

9.1 Vaatimusmäärittely

Vaatimusmäärittelyn onnistumista voidaan tarkastella itsenäisesti tai suhteessa arkkitehtuurin tekemiseen. Itsenäisenä työnä vaatimusmäärittely onnistui kohtuullisen hyvin, sillä vaatimukset saatiin dokumentoitua kattavasti ja sidosryhmät olivat tyytyväisiä vaatimusmäärittelyn tuloksiin.

Suhteessa arkkitehtuuriin vaatimusmäärittely oli vajaa laadullisten vaatimusten osalta. Laadullisten vaatimusten puuttuminen aiheutti ongelmia arkkitehtuurin määrittelyssä, mutta tätä ei osattu ennakoida vaatimusmäärittelyn aikana. Syynä laadullisten vaatimusten puuttumiseen voi olla se, että suunniteltu testauspalvelun on pohjimmiltaan yksinkertainen eikä vaadi monimutkaista arkkitehtuuria eikä myöskään laadullisten vaatimusten tarkkaa kartoitusta. Toinen mahdollinen syy on kokemuksen puute vaatimusmäärittelyn laatimisesta ja läpiviennistä.

Vaatimusmäärittelyssä oli tarkoitus kartoittaa testauspalvelun vaatimukset mahdollisimman laajasti ja toteuttaa näistä vaatimuksista aluksi vain tärkeimmät. Vaatimuksia ei kuitenkaan saatu kartoitettua kovin pitkälle tulevaisuuteen vaan vaatimukset ovat painottuneet testauspalvelun ensi vaiheen vaatimuksiin. Tälle voi olla syynä se, että sidosryhmät halusivat saada nopeasti konkreettisia tuloksia, sillä testauspalvelun odotettiin ratkaisevan monia ajankohtaisia ongelmia. Näin ollen vaatimusten kartoittamisen aikana ei katsottu tarpeeksi pitkälle tulevaisuuteen.

Vaatimusmäärittely toteutettiin useammassa iteraatiossa joista jokaisessa tarkastettiin siihen asti kerättyjen vaatimusten oikeellisuus. Vaatimusmäärittelyn valmistuttua olisi kuitenkin pitänyt tehdä vielä yksi, lopullinen läpikäynti jossa olisi käsitelty vaatimukset ja varsinkin niiden priorisointi. Tällainen lopullinen läpikäynti jäi tekemättä pääosin kiireen takia ja siksi, että tarvittavia henkilöitä oli vaikea saada yhteiseen tapaamiseen.

Vaatimusmäärittelyn käyttötapausten luonti aloitettiin liian myöhään. Käyttötapausten avulla olisi saatu sidosryhmiltä vaatimuksista enemmän palautetta. Nyt käyttötapauksia hyödynnettiin enimmäkseen arkkitehtuurin suunnittelussa.

Vaatimusmäärittely onnistui yleisesti ottaen hyvin, mutta kokemuksen vähyys työtavoista ja aikataulupaineiden luoma haastava ympäristö aiheuttivat vaatimusmäärittelyyn joitain puutteita. Tärkein kriteeri eli sidosryhmien tyytyväisyys kuitenkin täyttyi.

9.2 Arkkitehtuuri

Arkkitehtuuri ja toiminnallinen määrittely sisälsivät sellaiset tiedot joita suunnittelun alussa tavoiteltiin. Arkkitehtuurista saatiin looginen näkymä, prosessinäkymä sekä kuvaus rajapinnoista. Toiminnalliset määritykset kuvasivat testauspalvelun toimintaperiaatteen. Arkkitehtuurin suunnittelun tarkoituksena oli luoda sellainen arkkitehtuurisuunnitelma ja toiminnalliset määritykset, että testauspalvelun toteuttaminen voitaisiin aloittaa. Tästä saatiin vahvistus toteuttamisesta tarjouksen tehneeltä toimittajalta eli siltä osin arkkitehtuuri täytti vaatimukset.

Arkkitehtuurista ei tehty erillistä, lopullista arviointia. Näin ollen arkkitehtuuria, kuten vaatimusmäärittelyä, arvioidaan suunnitteluprosessin onnistumisen kannalta. Arkkitehtuurin suunnittelu seurasi tarkkaan ADD-prosessia ja suunnittelu eteni vaiheittain vaatimusmäärittelyn pohjalta. ADD on prosessina hyvin dokumentoitu ja esitetty ja sitä oli mahdollista seurata tarkasti ilman merkittävää kokemusta arkkitehtuurin suunnittelusta. Yksi isoimmista haasteista arkkitehtuurin suunnittelussa oli epätarkat laadulliset vaatimukset. ADD-prosessi vaatii toimiakseen riittävän tarkat laadulliset vaatimukset ja näiden puuttuminen aiheutti sen, että arkkitehtuuri on vajaa ainakin palvelun ulkoisten ja sisäisten rajapintojen osalta. Toinen merkittävä haaste oli, että arkkitehtuurin suunnittelun aikana verkkolaskutuksen liiketoiminnalliset ongelmat nousivat yhä enemmän esille. Näiden ratkaiseminen vei aikaa arkkitehtuurin suunnittelusta, mutta näitä ongelmia ja ratkaisuja ei sisällytetty tähän tutkimukseen.

Testauspalvelun arkkitehtuuria ja toiminnallisia määrityksiä voidaan arvioida myös kohdan 6.3 vaatimuslistaa vastaan. Testauspalvelu täyttää listan kohdat 3-9: sanoman tallentaminen, tallennetun sanoman käsittely ja haku, testitapauksen suorittamisen hallinta, sanoman lähettäminen, sanoman vastaanottaminen, sanoman sisällön validointi ja yhdenmukaisuustestien tulosten raportointi. Kohta 1, "Itsenäinen konfigurointi" täyttyy osittain, sillä testauspalvelu ei vaadi erillistä konfigurointia. Kohta 2, "ebXML-sanoman muodostaminen" tarkoittaa testauspalvelun tapauksessa verkkolaskun muodostamista. Tämä

kohta ei täyty sillä testauspalvelu ei osaa muodostaa itsenäisesti verkkolaskua vaan testauspalveluun on tallennettu valmiita malliverkkolaskuja.

Arkkitehtuurisuunnitelma ja toiminnalliset määritykset täyttivät niille asetetun tärkeimmän kriteerin eli testauspalvelun toteutuksen aloittamisen suunnitelmien pohjalta. Kuten vaatimusmäärittelyssäkin, kiire ja kokemuksen vähyys valitun menetelmän kanssa aiheuttivat joitain puutteita arkkitehtuuriin.

9.3 Tulosten luotettavuus

Luotettavuuden arviointi perustuu tutkimuksessa käytettyjen lähteiden luotettavuuden arviointiin. Tutkimuksessa käytetyt artikkelit olivat pääosin alan arvostetuista julkaisuista. Lähteinä käytetyt kirjat ovat yliopistoissa oppikirjoina ja kirjojen kirjoittavat tunnettuja alan tutkijoita. Tutkimuksessa suosittiin perinteisiä julkaisuja ja pyrittiin käyttämään mahdollisimman vähän puhtaasti www-pohjaisia lähteitä.

Tulosten luotettavuutta on vaikeaa arvioida objektiivisesti, koska tutkimuksen yhteydessä ei suoritettu kattavia arviointeja ja/tai läpikäyntejä. Sidosryhmät olivat tyytyväisiä lopulliseen vaatimusmäärittelyyn ja arkkitehtuuriin joten siltä osin ne täyttivät tehtävänsä. Mikäli testauspalvelu olisi myös toteutettu tutkimuksen aikana, voitaisiin tulosten luotettavuutta arvioida palvelun toimivuuden ja siitä saadun hyödyn perusteella.

Nykytilan tarkastelulle oli liian vähän aikaa, koska testauspalvelun suunnittelu oli pakko aloittaa nopeasti. Tämä aiheutti myös sen, että vastaavia töitä ei pystytty ottamaan testauspalvelun suunnittelussa huomioon tarpeeksi.

10 Johtopäätökset

10.1 Vaatimusmäärittely ja arkkitehtuuri

Tutkimus osoitti, että verkkolaskun testauspalvelu voisia toimia verkkolaskutuksen teknisten ongelmien ratkaisun apuna. Testauspalvelu on mahdollista toteuttaa siten, että useampi käyttäjä voi testata verkkolaskuja liiketoiminnallisia sääntöjä vastaan. Opiskelijaryhmän demo ja kirjallisuus osoittavat, että Schematron sopii XML-muotoisten sanomien testaamiseen, kun W3C XML Scheman ominaisuudet eivät riitä.

Sähköisen liiketoiminnan viitekehysten testaaminen kattaa viitekehysten kaikki osa-alueet: sanomat, prosessit ja viestinnän. Kattavan testauksen rakentaminen on kuitenkin haasteellista ja tässä tutkimuksessa tarkasteltu sanomien testaus on jo itsessään vaikea tehtävä. Sähköisen liiketoiminnan testauksen haasteellisuudesta ja lyhyestä historiasta kertoo toimivien testausalustojen vähäinen määrä.

Vaatimusmäärittely ja arkkitehtuurin suunnittelu osoittivat myös, että verkkolaskutuksen suurimmat ongelmat ovat liiketoiminnallisia eikä teknisiä. Verkkolasku on yksi ensimmäisistä uusien sähköisen liiketoiminnan viitekehysten sanomista. Muidenkin sanomien, kuten tilauksen, kanssa voidaan odottaa vastaavanlaisia liiketoiminnallisia ongelmia.

10.2 Verkkolaskutus yleisesti

Verkkolaskutuksen tilanne Suomessa on kaksijakoinen. Tilastojen valossa Suomi on maailman kärkimaita verkkolaskutuksessa ja kasvuvauhti on edelleen kova. Toisaalta liiketoiminnalliset ongelmat hidastavat ja vaikeuttavat verkkolaskutuksen käyttöönottoa ja tekevät verkkolaskutuksesta käyttäjien kannalta vähemmän houkuttelevaa. Käyttäjät haluaisivat verkkolaskutuksen toimivan kuten posti tai puhelinliikenne. Kirjeiden ja puhelujen välittäminen onkin hyvä rinnastus, sillä kummankin välitys on todennäköisesti alkanut vastaavanlaisesta tilanteesta jossa on useita toimijoita, useita toimintamalleja, sekavat hinnoittelu- ja sopimuskäytännöt. Puhelinliikenne Suomessa on lailla ja asetuksilla säädeltyä ja Viestintäviraston valvomaa toimintaa. Yhteentoimivuutta ei säädellä suoraan, mutta viestintävirasto tarkkailee mm. hintoja ja verkkovierailumaksuja sekä määrää korvauksia ongelmatilanteista. Näin puhelinliikenne on kilpailtua, mutta loppukäyttäjien kannalta kokonaisuus toimii.

Verkkolaskutuksen perusongelma on verkkolaskun kuuluminen kahteen eri maailmaan, sähköiseen liiketoimintaan ja sähköiseen maksuliikenteeseen. Verkkolaskutusta käsitellään näissä kahdessa eri maailmassa eri tavalla eri lähtökohdista. Tämä aiheuttaa sen, että yhteisistä asioista kuten välitysmalleista sopiminen on erityisen vaikeaa. Loppujen lopuksi sopimusneuvotteluissa on kysymys rahasta kuten liiketoiminnassa yleensäkin.

Verkkolaskutuksessa näyttäisi olevan nk. Prisoner's dilemma -tilanne. Tässä peliteorian asetelmassa kummankin osapuolen oman edun tavoittelu johtaa kokonaisuuden kannalta heikompaan tilanteeseen, kuin jos osapuolet olisivat tehneet myönnytyksiä toista osapuolta kohtaan. Tilanne missä useampi keskenään kilpaileva osapuoli yrittää sopia yhteisestä standardista on tuttu telemaailmasta, mm. Matkapuhelinstandardien puolelta. Tällaisten standardien luominen on haastavaa, sillä yrityksillä on liikesalaisuuksia ja omia liiketoiminnallisia intressejä jotka voivat olla ristiriidassa yhteisen sopimuksen kanssa.

Yhtenä ratkaisuna verkkolaskutuksen ongelmaan voisi olla yhtenäisten pelisääntöjen luominen jonkun viranomaisen toimesta. Tämä tarkoittaisi samalla kannanottoa kaikkea sähköistä sanomienvälitystä kohtaan. Kysymystä voi laajentaa edelleen: kuinka paljon Internetissä tapahtuvaa tiedonvälitystä tulisi säännellä viranomaisten toimesta?

Useamman verkkolaskuformaatin ongelma voidaan ratkaista luomalla yksi yleisformaatti, josta olisi muunnokset kaikkiin muihin formaatteihin. On epätodennäköistä, että edes pitkällä aikavälillä Suomessa olisi käytössä vain yksi ainut verkkolaskuformaatti. Vaikka tällainen formaatti saataisiin sovittua Suomessa, pitää myös ulkomaat ottaa huomioon. Yhden kaiken kattavan formaatin sijaan voitaisiin käyttää UBL:n (Universal Business Language) mallia, jossa minimitietosisällöstä räätälöidään maa-, toimiala- ja yrityskohtaisia versioita. Kaikkien näiden versioiden pohjautuminen samaan minimitietosisältöön takaisi räätälöityjen verkkolaskujen yhteentoimivuuden.

Sähköisen liiketoiminnan viitekehykset ja niiden testaus tulevat kehittymään tulevina vuosina. Viitekehysten jatkuvasti laajempi käyttö ja keskeisempi rooli liiketoiminnassa lisäävät kiinnostusta myös viitekehysten testaamiseen ja yhteentoimivuuden saavuttamiseen.

10.3 Jatkotutkimuskohteita

Tutkimuksen aikana nousi esille useita kysymyksiä joihin ei löytynyt vastauksia. Tässä käsitellään näitä kysymyksiä.

Sähköisen liiketoiminnan viitekehysten testaamisen osalta mielenkiintoinen kysymys on, minkälaisia tuloksia saadaan eri testaustavoilla ja kuinka hyvin teoriat tasoittain etenevästä testauksesta toimivat. Suositusten mukaan viitekehyksiä tulisi testata alkaen sanomamäärityksistä ja päätyen koko viitekehyksen testaamiseen aidossa ympäristössä, mutta kuinka hyvin tämä käytännössä toimii? Suosituksissa myös oletettiin, että standardit ja prosessit ovat tarkasti määritelty ennen testauksen aloittamista. Verkkolaskun tapauksessa näin ei ollut, joten olisiko verkkolaskun testaaminen pitänyt aloittaa jostain muusta tasosta kuin sanomista? Jos standardit ja toimintatavat ovat vasta muotoutumassa, mitä kannattaisi testata ensin?

Tässä tutkimuksessa ei käsitelty sitä prosessia, joilla verkkolaskun kaltaisia standardeja luodaan. Yhteisistä standardeista ja käytännöistä sopiminen tuo aina ongelmia, mutta ovatko ongelmat ja niiden ratkaisut samat sähköisen liiketoiminnan tapauksessa? Internetin standardeista on sovittu perinteisesti melko vapaalta ja kokeelliselta pohjalta. Miten tällainen toimintatapa sopii sähköisen liiketoiminnan sanomien, prosessien tai viitekehysten määrittämiseen?

Verkkolaskun testauspalvelun toimintamallia ei myöskään käsitelty tässä tutkimuksessa. Miten teknistä testauspalvelua tulisi käyttää apuvälineenä muiden kuin teknisten ongelmien ratkaisemiseksi? Miten testauspalvelun testien kehittäminen tulisi organisoida siten, että testauspalvelu voisi samalla kehittää verkkolaskuformaatteja eteenpäin? Formaatin muutokset tulevat yleensä käyttäjiltä, jotka esittävät uusia ja/tai muuttuneita tarpeita. Formaatin uuden version toteutus ohjelmistoihin on kuitenkin ohjelmistotalojen vastuulla. Voisiko testauspalvelu yhdistää käyttäjät, ohjelmistotalot ja formaattien kehittäjät siten, että kehitysprosessia voitaisiin nopeuttaa?

Viimeinen ehdotus jatkotutkimuskohteeksi liittyy rakennekuvausten käyttöön testaamisessa. Miten useamman eri rakennekuvauskielen mukaisten rakennekuvausten ylläpito tulisi tehdä, jotta rakennekuvaukset säilyvät ymmärrettävinä?

11 Viitteet

Bachmann F., Bass L., Chastek G., Donohoe P., Peruzzi F., 2000, "The Architecture Based Design Method", CMU/SEI-2000-TR-001, Carnegie Mellon, Pittsburgh, http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr001.pdf

Bachmann, F., Bass, L., 2001, "Introduction to the attribute driven design method", Proceedings of the 23rd International Conference on Software Engineering, s.745 – 746

Bass, L., Clements, P., Kazman, R., 2003, "Software architecture in practice", 2nd edn, Addison-Wesley, Boston, ISBN: 0-321-15495-9

Bass, L., Klein, M., Bachmann, F., 2001, "Quality Attribute Design Primitives and the Attribute Driven Design Method", Proceedings of the Product Family Engineering, vol. 4. Springer-Verlag, Berlin

Bochmann G., Petrenko A., 1994, "Protocol testing: review of methods and relevance for software testing", Proceedings of the 1994 international symposium on Software testing and analysis, pp.109-124, Seattle, Washington, United States

Boehm, B., Basili, V.R., 2001, "Software Defect Reduction Top 10 List", Computer, vol. 34, Issue 1, s. 135-137

Booch, G., Rumbaugh, J., Jacobson, I., 1998, "The unified modeling language user guide", Addison-Wesley, Reading (MA), ISBN 0-201-57168-4

Burnstein, I, 2003, "Practical software testing: a process-oriented approach", New York, Springer

Bussler, C., 2002, "Modeling and Executing Semantic B2B Integration", Proceedings of Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, RIDE-2EC 2002, pp. 69 - 74

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Satfford, J., 2003, "Documenting software architectures : views and beyond", Boston : Addison-Wesley, cop. 2003, ISBN: 0-201-70372-6

Durand, J., Kass, M., Wenzel, P., 2003, "The ebXML Test Framework and the Challenges of B2B Testing", XML Europe 2003

eInvoice Consortium, 2005, "Suomalaisen verkkolaskuoperaattorin tilitunnisteen rakenne", http://www.einvoiceconsortium.com/pdf/verkkolasku-account.pdf, (luettu 20.8.2005)

Elma, 2005a, "Elma eInvoice", http://www.elma.net/fi/tuotteet/verkkolasku/einvoice/, (luettu 29.08.2005)

IBM, 2005, "Rational Unified Process", http://www-306.ibm.com/software/awdtools/rup/ (luettu 1.6.2005)

IDC, 2003, " B2B Electronic Invoicing Market in Finland", http://www.tieke.fi/mp/db/material_folder/x/IMG/15394:15069/file/Salmi_Paperitontaloushallintotanaan-meillajamuualla.pdf

IEEE (Institute of Electrical and Electronics Engineers), 1990, "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries", New York, NY

ISO/IEC, 2004, "Guide 2:2004 - Standardization and related activities -- General vocabulary"

Karlsson, J., Wohlin, C. and Regnell, B. (1998). "An evaluation of methods for prioritizing software requirements", Journal of Information and Software Technology, vol. 39, no. 14-15, s. 939-947.

Kim, D., Yun, J-H., 2003, "Development of an ebXML Conformance Test System for e-Business Solutions", Proceedings of 4th International Conference on E-Commerce and Web Technologies, pp. 145-154

Kruchten, P, 2004, "The rational unified process : an introduction", Addison-Wesley, ISBN: 0-321-19770-4

Kruchten, P., B., 1995, "The 4+1 View Model of Architecture", IEEE Software, November 1995, Volume 12, no. 6, s. 42-50, ISSN:0740-7459

Kulvatunyou, B., Ivezic, N., Martin, M., Jones, A., 2003, "A Business-to-Business Interoperability Testbed: An Overview", Proceedings of the 5th international conference on Electronic commerce

Lawrence, B., 1996, "Requirements happens…", American Programmer, vol. 10, no. 4, p. 3-9

Lee, D., Chu, W, 2000, "Comparative Analysis of Six XML Schema Languages", ACM SIGMOD Record Vol. 29, Issue 3, pp. 76–87

Lee, H., Whang, S., 2001, "E-Business and Supply Chain Integration", Stanford Global Supply Chain Management Forum SGSCMF- W2-2001

Li, E. Y., 2003, "From E-Commerce to E-Business", International Journal of Electronic Business, vol. 1, no. 1, pp. 1-2

Lim, B., Wen, H., 2002, "The impact of next generation XML", Information Management & Computer Security, vol.10, no. 1, pg. 33

Market-Visio Oy, 2005, "Verkkolaskujen käyttö Suomessa"

Medvidoc, N., Rosenblum, D., S., Fredmiles, D., F., Robbins, J., E., 2002, "Modeling software architectures in the Unified Modeling Language", ACM Transactions on Software Engineering and Methodology (TOSEM), January 2002, vol. 11, no. 1, s. 2-57, ISSN:1049-331X

Myers, G., 1979, "The Art of Software Testing", Wiley-Interscience

NIST, 1999, "Interoperability Cost Analysis of the U.S. Automotive Supply Chain", (Planning Report #99-1)

NIST, 2005a "NIST", http://www.nist.gov, luettu 15.9.2005

NIST, 2005b, "Semantic Checking Documentation", http://syseng.nist.gov/b2bTestbed/projects/semanticChecking/UML/html/root.html, luettu 14.5.2005

OASIS, 2005, "IIC ebXML Test Framework V1.1 Technical Committee Specification"

OES (Danish Agency for Governmental Management), 2005, http://www.oes.dk/sw1903.asp, (luettu 20.8.2005)

Riggs, S., 2003, "Data quality and XML validation", Proceedings of XML Europe 2003, www.idealliance.org/papers/xmle03/slides/riggs/riggs.ppt (poista url)

Rodgers, J., Yen, D., Chou, D., 2002, "Developing E-Business: a Strategic Approach", Information Management & Computer Security, vol. 10, no. 4, pp. 184–192.

Rosenthal, L., Brady, M., 2000, "Conformance", http://www.itl.nist.gov/div897/ctg/conformance/xml2000-conformance.ppt

RosettaNet, 2005, http://www.rosettanet.org, (luettu 15.7.2005)

SFTI, 2005, "Svefaktura – Online verifiering", http://www.svefaktura.se/, luettu 15.10.2005

Saaty, T.L., 1980, "The Analytic Hierarchy Process", McGraw-Hill, Inc.

Schematron, 2005, http://www.schematron.com, (luettu 26.8.2005)

Shim, S., Pendyala, V., Sundaram, M., Gao, J., 2000, "Business-to-Business E-Commerce Frameworks", IEEE Computer, vol. 33, no. 10, pp. 40–47

Sisäasiainministeriö, 2003, " Verkkolaskujen käyttö julkishallinnossa (JHS155)"

Sommerville, I. and Sawyer, P. 1997. "Requirements Engineering – A Good Practice Guide", John Wiley & Sons, New York, ISBN:

Sommerville, I., 2004, "Software Engineering", Harlow, Addison-Wesley, 7th edition, 963 s., ISBN: 0-321-21026-3

Soni, D., Nord, R. L., Hofmeister C., 1995, "Software architecture in industrial applications", Proceedings of the 17th International Conference on Software Engineering, s. 196-207

Statistics Denmark, Statistics Finland, Statistics Norway, Statistics Sweden, 2001, "Use of IC in Nordic Enterprises", Statistics Norway, Kongsvinger, Norway

Suomen Pankkiyhdistys, 2005a, "Finvoice - verkkolasku", http://www.pankkiyhdistys.fi/verkkolasku, (luettu 15.7.2005)

Suomen Pankkiyhdistys, 2005b, "Finvoice soveltamisohje / Versio 1.2", http://www.pankkiyhdistys.fi/verkkolasku/soveltamisohje12.pdf, luettu 20.4.2005

TIEKE Tietoyhteiskunnan kehittämiskeskus, 2005, http://www.tieke.fi/tuotteet_ja_palvelut/tietoa_verkkolaskusta, (luettu 15.5.2005)

TIEKE, 2005a, "Tietoa verkkolaskusta", http://www.tieke.fi/tuotteet_ja_palvelut/tietoa_verkkolaskusta, (luettu 10.6.2005)

TIEKE, 2005b, "Verkkolaskufoorumi", http://www.tieke.fi/verkostot/verkkolaskufoorumi/

TIEKE, 2005c, "Verkkolaskuosoitteisto", http://verkkolasku.tieke.fi/, (luettu 15,8,2005)

TIEKE, 2005d, "Minimitietosisältötaulukko", http://www.tieke.fi/mp/db/file_library/x/IMG/13019/file/Mappaustaulukko.pdf, (luettu 29.8.2005)

TietoEnator, 2005, "TEAPPSXML – verkkolasku", http://www.tietoenator.fi/default.asp?Path=408,410,16094,1129,18031,18138,8259,10178,12751, (luettu 15.7.2005)

Vahtera, P., Salmi, H., 1997, "Internet and EDI in Effective Accounting"

Valtiokonttori, 2005, Keskustelut Kim Hacklinin ja Kristiina Seppälän välillä syksy 2005

W3C, 2004, "Extensible Markup Language (XML) 1.0 (Third Edition)", http://www.w3.org/TR/2004/REC-xml-20040204/, (luettu 17.8.2005)

Wallentine, V., Zhou, S., 2002, "Validating XML document content with the object constraint language", In Proceedings of the 2002 Internet Conference, Las Vegas, NY

Warren, P., Reakes, G., Massari, A., 2003, "Business rules validation – the standard the W3C forgot", Proceedings of XML Europe 2003, http://www.idealliance.org/papers/dx_xmle03/papers/03-02-03/03-02-03.pdf (poista url)

Wiegers, K. 2003. "Software Requirements", Microsoft Press, Redmond Washington, ISBN:0-7356-1879-8